VŠB – Technická univerzita Ostrava

Fakulta strojní

Katedra automatizační techniky a řízení

Tvorba úloh pro LEGO Mindstorms v .NET Micro Framework

Programming Tasks for LEGO Mindstorms using .NET Micro Framework

Student: Bc. David Čermák

Vedoucí diplomové práce: Ing. David Fojtík, Ph. D.

Ostrava 2015

Vysoká škola:	VŠB – Technická univerzita Ostrava	Fakulta:	strojní
Katedra:	automatizační techniky a řízení 352	Školní rok:	2014/2015

ZADÁNÍ DIPLOMOVÉ PRÁCE

posluchač: **BC. DAVID ČERMÁK**

inženýrský obor: Automatické řízení a inženýrská informatika

číslo: 3902T004-00

vedoucí diplomové práce: Ing. David Fojtík, Ph.D.

Název tématu: **Tvorba úloh pro LEGO Mindstorms v .NET Micro Framework**

Programming Tasks for LEGO Mindstorms using .NET Micro Framework

Zásady pro vypracování:

1. Seznamte se s platformou .NET Micro Framework a .NET Gadgeteer a vývojovou deskou FEZ Cerbuino Bee/NET.
2. Seznamte se se stavebnicí LEGO Mindstorms education NXT 2.0 a NXT 3.0 a popište způsob jejich programování, dostupné senzory a pohony.
3. Seznamte se s vývojovou deskou NXShield-Dx a s možnostmi programování modelů stavebnice LEGO Mindstorms Education na platformě .NET Micro Framework.
4. Na platformě .NET Micro Framework vytvořte demonstrující úlohy, které využívají vybrané pohony a senzory stavebnice LEGO Mindstorm Education.
5. Zhodnoťte dosažené výsledky a navrhněte směry dalšího řešení.

Seznam odborné literatury:

MALIN, R. John a Sean LIMING. Professional's Guide to .NET Micro Framework Application Developement. Yorba Linda, CA, USA: Annabooks, 2012. ISBN 978-0-9842801-9-3.

JONES, Allen. C# programmer's cookbook. Redmond, Wa.: Microsoft Press, c2004, xvii, 628 p. ISBN 0735619301.

VALK, Laurens. The Lego Mindstorms Ev3 Discovery Book: A Beginner's Guide to Building and Programming Robots. San Francisco: No Starch Press, 2014. 396 s. ISBN 1-59327-532-3.

FARANA, R., SMUTNÝ, L., VÍTEČEK, A. 1999. Zpracování odborných textů z automatizace a informatiky. 1. vyd. Ostrava: VŠB-TU Ostrava, 1999. 68 s. ISBN 80-7078-737-6.

Datum zadání: 13.12.2014

Datum odevzdání: 18.05.2015

Vedoucí diplomové práce: **Ing. David Fojtík, Ph.D.**

Prohlašuji, že jsem celou diplomovou práci včetně příloh vypracoval samostatně pod vedením vedoucího diplomové práce a uvedl jsem všechny použité podklady a literaturu.

V Opavě 18.5.2015 .....

Prohlašuji, že

- jsem byl seznámen s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., autorský zákon, zejména § 35 – užití díla v rámci občanských a náboženských obřadů, v rámci školních představení a užití díla školního a § 60 – školní dílo.
- beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen„VŠB-TUO“) má právo nevýdělečně ke své vnitřní potřebě diplomovou práci užít (§ 35 odst. 3).
- souhlasím s tím, že diplomová práce bude v elektronické podobě uložena v Ústřední knihovně VŠB-TUO k nahlédnutí a jeden výtisk bude uloženu vedoucího diplomové (bakalářské) práce. Souhlasím s tím, že údaje o kvalifikační práci budou zveřejněny v informačním systému VŠB-TUO.
- bylo sjednáno, že s VŠB-TUO, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu § 12 odst. 4 autorského zákona.
- bylo sjednáno, že užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití mohu jen se souhlasem VŠB-TUO, která je oprávněn a v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů,které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše).
- beru na vědomí, že odevzdáním své práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, bez ohledu na výsledek její obhajoby.

V Opavě : 18.5.5015 .....
podpis

Bc. David Čermák

Česká 1131/9
Opava 6
74706

ANOTACE DIPLOMOVÉ PRÁCE

ČERMÁK, D. *Tvorba úloh pro LEGO Mindstorms v .NET Micro Framework: diplomová práce.* Ostrava: VŠB - Technická universita Ostrava, Fakulta strojní, Katedra automatizační techniky a řízení, 2015, 75 s. Vedoucí práce: Fojtík, D.

Diplomová práce je zaměřena na aplikaci metody nahrazení logické kostky stavebnice LEGO Mindstorms Education vývojovou deskou NXShield-Dx v kombinaci s jednodeskovým počítačem FEZ Cerbuino Bee. NXShield-Dx slouží pro zapojení jednotlivých modulů stavebnice LEGO Mindstorms a FEZ Cerbuino Bee prostřednictvím navrženého software čte data naměřená LEGO senzory, nastavuje řídicí parametry LEGO servomotorů a umožňuje využívat zapojené Gadgeteer moduly. Komunikace je zprostředkována v rámci protokolu I^2C. Vytvořený program je napsán v jazyce C# v rámci .NET Gadgeteer projektu a obsahuje naprogramované demonstrativní úlohy využívající zároveň LEGO Mindstorms Education i Gadgeteer moduly.

KLÍČOVÁ SLOVA

.NET Gadgeteer, .NET Micro Framework, Arduino sběrnice, C#, FEZ Cerbuino Bee, Gadgeteer slot, I^2C, LEGO Mindstorms Education, NXShield-Dx

ANNOTATION OF MASTER THESIS

ČERMÁK, D. *Programming Tasks for LEGO Mindstorms using .NET Micro Framework: Master Thesis.* Ostrava: VŠB - Technical University of Ostrava, Faculty of Mechanical Engineering, Department of Control Systems and Instrumentation, 2015, 75 p. Thesis head: Fojtík, D.

This thesis is focused on application of method to replace LEGO Mindstorms Education Logic Brick with development board NXShield-Dx in combination with single-board computer FEZ Cerbuino Bee. NXShield-Dx is used for connecting the LEGO Mindstorms modules and FEZ Cerbuino Bee reads data measured by the LEGO sensors, sets the control parameters of the actuators and enables the use of connected Gadgeteer modules through the developed software functions. Communication is realized on I2C protocol basis. The program is written in C # programming language as .NET Gadgeteer project and includes demonstrative tasks which are using LEGO Mindstorms Education modules and also Gadgeteer modules.

KEYWORDS

.NET Gadgeteer, .NET Micro Framework, Arduino Headers, C#, FEZ Cerbuino Bee, Gadgeteer slot, I^2C, LEGO Mindstorms Education, NXShield-Dx

Obsah

Seznam použitých termínů, zkratek a symbolů

1-Wire	Komunikační protokol
ARM	Advanced RISC Machine
b	Bit
B	Byte
BMP	Bitmap
Break point	Bod zastavení chodu programu při ladění
CAN	Controller Area Network
CPU	Central Processing Unit
Datasheet	Dokumentace elektronického zařízení
e	Regulační odchylka měřené veličiny od žádané hodnoty
Ethernet	Souhrn technologií protokolů LAN.
Firmware	Software vestavěného systému.
Garbage collection	Metoda automatické správy paměti
GUI/UI	Graphical User Interface/ User Interface
GPIO	General Purpose Input/Output
Hardware	Fyzický komponent výpočetní techniky
Chipset	Zařízení s více spolupracujícími integrovanými obvody
In-Field Update	Přepsání paměti programu při zachování jeho chodu
LCD	Liquid Crystal Display
LED	Light Emission Diode
Linker	Program spojující objektové soubory do jednoho souboru
LSB	Least Significant Bit
MMU	Memory Management Unit
MSB	Most Significant Bit
Namespace	Jmenný prostor slučující třídy programovacího jazyka
Open-source	Software s licenčně přístupným kódem
Port	Přepsání software pro přenesení na jinou platformu
PWM	Pulse Width Modulation
RAM	Random Acess Memory
RISC	Reduced Instructions Set Computing
s	Sekunda

SCL	Serial Clock
SDA	Serial Data
SDK	Software Developement Kit
Serializace	Převod datové struktury na bitový řetězec
Socket	Rozhraní pro zapojení periferie.
SPI	Serial Periphetial Interface
SSL	Secure Sockets Layer
TCP/IP	Transmission Control Protocol/Internet Protocol
u	Akční veličina
UART	Universal asynchronous receiver/transmitter
USB	Universal Serial Bus
y	Měřená veličina
w	Žádaná hodnota měřené veličiny

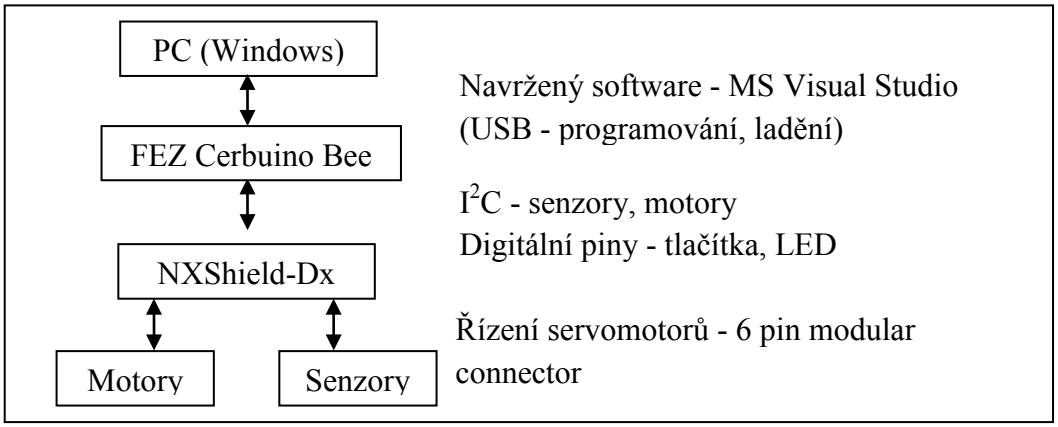
1 Úvod

Tématem diplomové práce je návrh software pro jednodeskový počítač FEZ Cerbuino Bee, který bude prostřednictvím vývojové desky NXShield-Dx zajišťovat řízení modulů stavebnice LEGO Mindstorms Education v rámci navržených demonstrativních úloh. Software je také vhodné vytvořit s ohledem na využití Gadgeteer modulů, pro jejichž použití byl jednodeskový počítač navržen. V aplikaci je potřeba užití základních senzorů a aktuátorů ze stavebnice LEGO Mindstorms Education a řídicí programovatelná jednotka Logic Brick z této stavebnice je nahrazena jednodeskovým počítačem FEZ Cerbuino Bee, který přistupuje k modulům prostřednictvím vývojové desky NXShield-Dx. Na obrázku 1.1 je schematicky znázorněn princip interakce jednotlivých zařízení.

Jednotlivá témata, jenž jsou obsahem tohoto dokumentu, jsou v návaznosti k problematice přístupu k funkcím modulů stavebnice LEGO Mindstorms prostřednictvím jednodeskového počítače FEZ Cerbuino Bee komunikujícího s vývojovou deskou NXShield-Dx. Nejprve je pozornost věnována seznámení s vývojovými platformami .NET Micro Framework, .NET Gadgeteer a vývojovou deskou FEZ Cerbuino Bee. Prostřednictvím zmíněných platforem lze naprogramovat, zkompilovat a nahrát software do paměti jednodeskového počítače.

Dále jsou blíže popsány základní zařízení stavebnice i s různými metodami tvorby programu pro počítač LEGO Mindstorms Education Logic Brick. Zmíněný řídicí počítač stavebnice LEGO Mindstorms je v uvažované úloze nahrazen vývojovou deskou NXShield-Dx, která zastává funkci řídicí jednotky a pro dosažení požadovaného řízení je nutné uvést metody jejího ovládání. Přístup k vývojové desce je zprostředkován v rámci komunikace I^2C, je tedy nutné blíže rozvést princip tohoto protokolu a jednotlivé komunikační parametry vývojové desky.

V rámci tvorby demonstrativních úloh je nejprve vhodné definovat jednotlivé funkce pro operace se zařízeními stavebnice LEGO Mindstorms a poté za pomocí jejich užití realizovat algoritmy plnící navržené úlohy.



Obr. 1.1 Schéma procesu od programování jednodeskového počítače po ovládání jednotlivých modulů stavebnice LEGO Mindstorms Education

2 Seznámení s použitými vývojovými platformami

Návrh zmíněného software pro jednodeskový počítač FEZ Cerbuino Bee je uvažován prostřednictvím vývojového software MS Visual Studio 2012. Program je vyvíjen v programovacím jazyce C# jako projekt vývojové platformy .NET Gadgeteer, která je založena na platformě .NET Micro Framework. V této kapitole jsou rozvedeny možnosti a metody užití zmíněných platforem a následně jsou uvedeny vlastnosti jednodeskového počítače FEZ Cerbuino Bee.

2.1 Platforma .NET Micro Framework

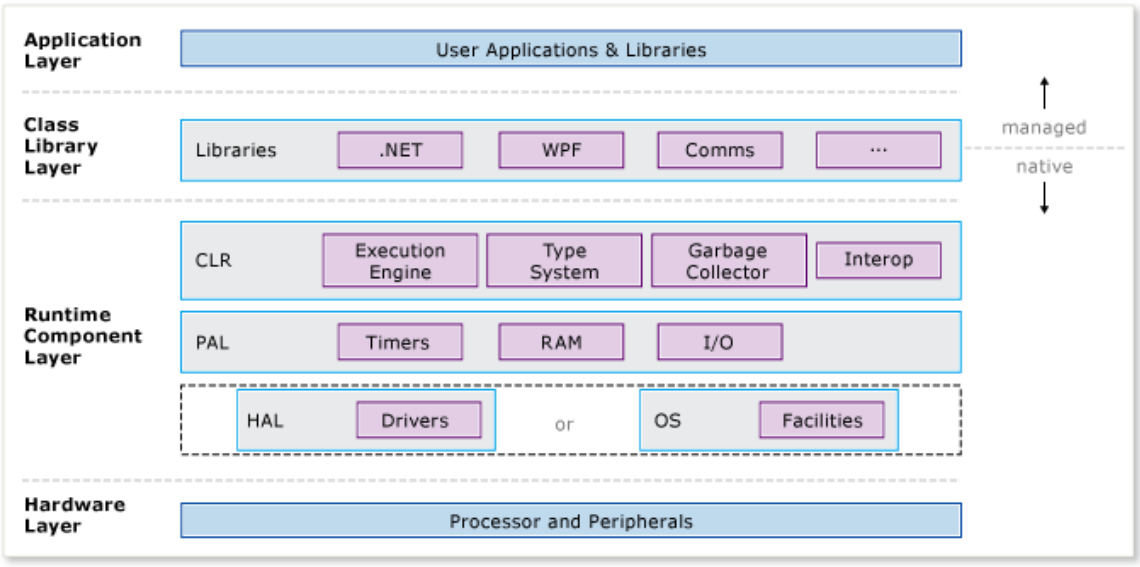
.NET Micro Framework je open source platforma, navržena na operační systém Windows, která rozšiřuje možnosti využití vývojové platformy .NET společnosti Microsoft. Vytvořený software prostřednictvím této platformy slouží pro ovládání malých vestavěných zařízení a modulů (Microsoft Open Technologies, 2011). Prostředky a metody užívané při programování v Microsoft .NET mohou být s výhodou využívány a přeneseny na tuto platformu. Zařízení, pro které je vyvíjen software prostřednictvím platformy .NET Micro Framework jsou většinou jednodeskové počítače osazeny 32 bitovým procesorem. Tyto jednodeskové počítače nemusí mít podporu správy paměti (MMU) a velikost paměti RAM může nabývat malých kapacit již od 64 Kb.

Jednou z předních výhod využití platformy .NET Micro Framework je možnost vývoje software v jednom z mnoha vyšších programovacích jazyků. Při programování ve vyšším programovacím jazyku, jako je například jazyk C#, podporovaný platformou .NET Gadgeteer, je možné využít předprogramovaných funkcí a metod, které umožňují rychlejší a přehlednější zápis programu. U zařízení používaných pro aplikace se složitými protokoly, jako například TCP/IP protokol pro síťovou komunikaci, je bezpečnější a rychlejší používat funkce vyššího programovacího jazyka, protože jsou odladěny, je k nim většinou dostupná dokumentace a ověřené metody použití.

Součástí zadání diplomového projektu je návrh software pro vývojovou desku FEZ Cerbuino Bee, ke které je připojena řídicí jednotka NXShield-Dx ovládající zvolené pohony a senzory ze stavebnice LEGO Mindstorms Education. Zmíněna řídicí jednotka je náhradou za programovací jednotku stavebnice LEGO Minsdstorms, kdy zastává pouze funkce řízení zapojených modulů a komunikace s programovatelnou jednotkou. Vlastní program běží na počítači FEZ Cerbuino Bee a s řídicí jednotkou komunikuje prostřednictvím protokolu I^2C. Tato problematika komunikace je dále blíže rozebrána v rámci tvorby demonstrativních úloh. Jednotlivé následující informace o platformě .NET Micro Framework jsou převzaty ze stránek společnosti Microsoft pro vývojáře (Microsoft, 2015).

Architektura .NET Micro Framework

Architektura .NET Micro Framework se dělí na dvě hlavní části, softwarovou a hardwarovou architekturu. Následující schéma představuje shrnutí jednotlivých vrstev těchto architektur.



Obr. 2.1 Schéma architektury .NET Micro Framework

Dále jsou blíže popsány jednotlivé vrstvy architektur, kdy je vhodné začít od "nejnižší" vrstvy, kterou je vrstva Hardwarová.

Hardwarová vrstva (Hardware Layer)

Předpokladem je, že hardwarová vrstva sestává z pevných komponent uvažované aplikace, tedy procesoru a zapojených periferií nebo libovolných zvolených elektrických obvodů. Tato vrstva představuje hardwarové komponenty, které jsou v rámci .NET Micro Framework podporovány nebo je k jejím vstupům/výstupům možné prostřednictvím platformy přistupovat.

Množství podporovaných zařízení stále narůstá. Jedná se především o hardware navržený pro operační systém Microsoft Windows nebo pro systémy společností spolupracující se společností Microsoft. S výhodou je ovšem také možný port .NET Micro Framework i na jiný chipset, který není aktuálně podporovaný.

Vrstva runtime komponent (Runtime Component Layer)

Tato vrstva sestává ze tří komponent, Common Language Runtime, Hardware Abstraction Layer a Platform Abstraction Layer.

- HAL, PAL - Tyto dva komponenty RCL vrstvy jsou základem pro plnění softwarových C++ funkcí s použitým hardwarem. Funkce PAL jsou nezávislé na použitém hardwaru a při jeho záměně je není potřeba modifikovat. Rozdílem jsou HAL komponenty, které je potřeba navrhnout s ohledem na použitý hardware.

- CLR - Jedná se o podmnožinu prostředků, které zajišťují chod ohledně správy paměti, spouštění vláken jednotlivých procesů, kódů a jiných systémových služeb. Velkou předností je velmi malá paměť potřebná pro zajištění všech podporovaných funkcí (okolo 390 KB (Microsoft, 2015)). Jednotlivé funkce .NET Micro Framework CLR jsou seskupeny dle typu v následující tabulce 2.1.

Tab. 2.1 Funkce komponentu CLR platformy .NET Micro Framework

Funkce CLR .NET Micro Framework
Číselné typy, typy tříd, typy polí (pouze jednorozměrné), události, reference
Synchronizace, správa procesních vláken, časovače
Serializace
Garbage collection
Správa vyjímek (např.: prevence chyb, maximalni doba běhu jednotlivých vláken, interní reprezentace stringů jako Unicode UTF-8, výskyt vícerozměrných polí)

Vrstva knihoven tříd (Class Library Layer)

Jedná se o objektově orientovaný soubor tříd, obsahující knihovny umožnující:

- šifrování
- ladění aplikace, grafické prvky
- přístup k vytvořeným C# aplikacím, podporující rozšíření funkce chipsetu o například sériovou komunikaci, SPI, I^2C
- knihovny shellu, CLR API a corelib

Aplikační vrstva

Tato vrstva je nejvyšší z uvedených vrstev a obsahuje navržené aplikace pro zvolená zařízení. Typ aplikace se vždy odvíjí od použitého hardware. Jako jediný podporovaný jazyk touto aplikační vrstvou je jazyk C#.

2.2 Platforma .NET Gadgeteer

Platforma .NET Gadgeteer je open-source platformou rozšiřující možnosti použití .NET Micro Framework při vývoji software pro aplikace malých elektronických přístrojů. Umožňuje uživateli poměrně snadnější a rychlejší programovaní u podporovaných elektronických zařízení, než je tomu u samotné platformy .NET Micro Framework. Podporované moduly kompatibilní s podporovanými zařízeními, jako jsou například senzory, světla, přepínače, displeje, komunikační moduly a další, jsou připojeny k vývojové desce pomocí standardizovaných sběrnic (GHI Electornics LLC, 2015), které jsou kompatibilní s platformou .NET Gadgeteer a říká se jim "Gadgeteer headers". Další výhodou je množství funkcí, sloužících pro komplexní úlohy ovládání podporovaných modulů. Při programování aplikace pro práci s přístroji od GHI Electronics má programátor možnost použít funkce obsažené v nástroji .NET Gadgeteer, pomocí kterých lze ovládat a využívat veškeré funkce jednotlivých periferií.

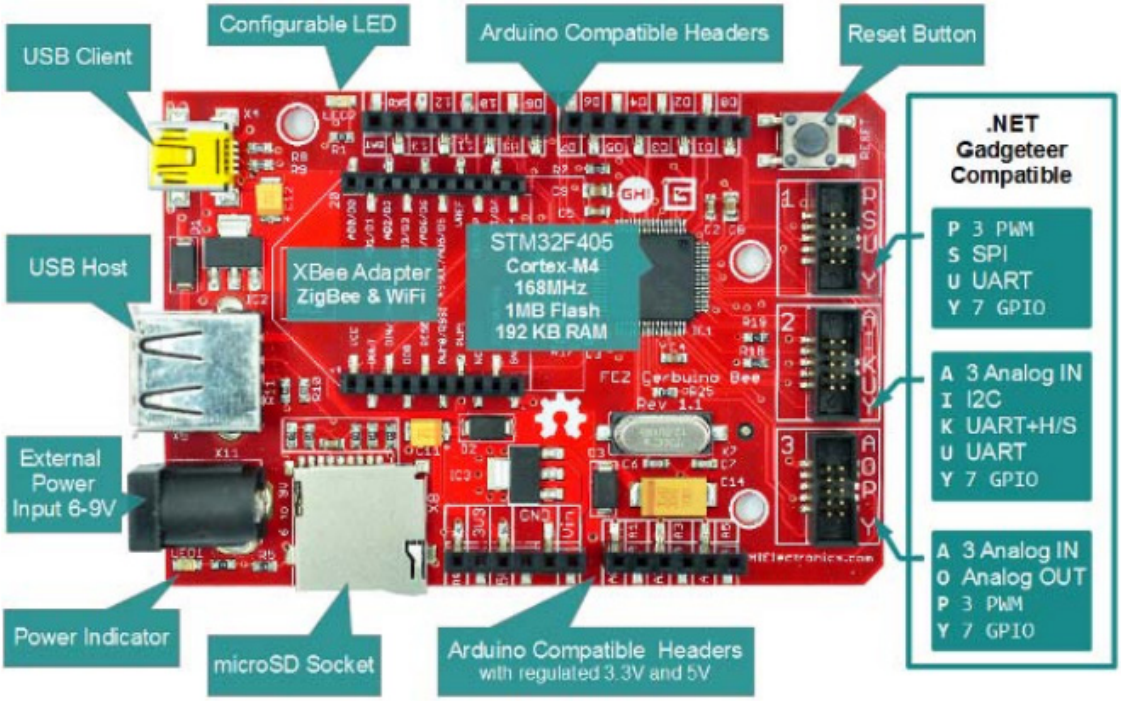
Platforma .NET Gadgeteer je tedy nejvhodnější volbou vývojového prostředí, která je pro vývoj demonstračního programu k dispozici. S výhodou je toto prostředí využito, protože obsahuje nástroje pro vlastní kompilaci a následné nahrání programu do paměti jednodeskového počítače FEZ Cerbuino Bee, kdy umožňuje uživateli využívat všechny prvky platformy .NET Micro Framework i metody pro komunikaci s moduly .NET Gadgeteer.

2.3 Vývojová deska FEZ Cerbuino Bee/NET

FEZ Cerbuino Bee je jednodeskový počítač podporovaný platformou .NET Gadgeteer a uživatel má pro něj možnost programovat aplikace prostřednictvím software Visual Studio v programovacím jazyku Visual Basic nebo C#. Ve zmíněném vývojovém prostředí je možné využívat velké množství usnadnění v podobě knihoven s předprogramovanými funkcemi pro aplikace využívající síťové komunikace, systémy pro správu souborů, grafické rozhraní a periferie (GHI Electronics LLC, 2013). Obrázky a informace, uvedené v následujících podkapitolách, jsou převzaty z datasheetu výrobce vývojové desky (GHI Electronics LLC, 2012).

2.3.1 Hardware FEZ Cerbuino Bee

Pro bližší představu o rozhraní jednodeskového počítače je převzata fotografie z datasheetu výrobce, kde jsou jednotlivá rozhraní vyznačena. Význam jednotlivých komponent je dále blíže popsán. Po jejich představení se lze věnovat jednotlivým hlavním parametrům a funkcím, které lze u jednodeskového počítače využít.



Obr. 2.2 Fotografie vývojové desky FEZ Cerbuino BEE s popisem jednotlivých komponent, vyňatá z datasheetu výrobce (GHI Electronics LLC, 2012)

External Power Input 6-9V

Slouží pro přívod napájecího napětí v rozmezí od 6V do 9V.

Power Indicator

Jedná se o indikační LED diodu, která při rozsvícení uživateli předává informaci o správném napájení desky prostřednictvím External Power Input 6–9V. Pokud není zmáčknuto tlačítko RESET a uvedená dioda nepřerušovaně svítí, znamená to, že je vykonáván program nahraný v paměti počítače.

MicroSD socket

MicroSD socket slouží pro připojení MicroSD paměťové karty, k jejíž obsahu je dále možné přistupovat.

Arduino compatible headers

Tyto sběrnice sestávají celkem z 28 PINů a slouží pro připojení modulů, které jsou kompatibilní s elektronickou platformou Arduino. Lze na něj také připojit libovolný komponent, pro který je však potřeba vytvořit knihovnu (Arduino, 2007). V uvažované aplikaci řízení modulů stavebnice LEGO Mindstorms je k použitému jednodeskovému počítači FEZ Cerbuino Bee připojena logická kostka NXShield-Dx prostřednictvím zmiňované sběrnice.

.NET Gadgeteer Compatible Sockets

Tyto sockety slouží pro připojení jednotlivých modulů společnosti GHI Electronics, které jsou určeny specificky pro zapojení do těchto slotů. Každý z podporovaných modulů nelze zapojit do libovolného socketu, protože každému z těchto tří konektorů jsou přístupny jiné signály. Proto jsou u každého slotu vytištěny značky v podobě písmen, které určují, zda lze daný komponent do tohoto socketu zapojit a následně používat. Značení na modulu musí být tedy vyznačeno také u socketu, do kterého jej zapojujeme. Při vývoji software prostřednictvím platformy .NET Gadgeteer je také nutné zapojení nastavit v grafickém rozhraní vývojového prostředí. Význam jednotlivých značek je uveden na obrázku 2.2.

Reset Button

Toto tlačítko slouží pro restartování počítače, při jeho zmáčknutí je vymazána paměť RAM a běžící program je zastaven. Po vrácení tlačítka do původního stavu počítač nahraje do paměti program a začne jej opět vykonávat ze začátku.

Configurable LED

Ke stavu této světelné diody je možné přistupovat a změnit dle požadavku její stav mezi vypnuto/zapnuto na nastavenou dobu.

USB Client

Tento port USB je dle výrobce určen pouze pro ladění a nahrání programu do pevné paměti počítače. Prostřednictvím tohoto portu lze jednodeskový počítač napájet.

USB Host

Tento slot slouží pro připojení jiného elektronického zařízení prostřednictvím USB a lze jej také tímto portem napájet.

XBee Adapter

Do této sběrnice lze zapojit libovolný XBee modul.

CPU Cortex-M4

Hlavní procesní jednotka Cortex-M4 je výpočetním prvkem počítače. Součástí této výpočetní jednotky je také vlastní FLASH a RAM paměť. Architektura tohoto procesoru je typu ARM, díky které má procesor za chodu poměrně nízkou spotřebu elektrické energie a jednodeskový počítač je tedy vhodný při využití v mobilních aplikacích.

2.3.2 Parametry FEZ Cerbuino Bee

Parametry jednodeskového počítače FEZ Cerbuino BEE jsou shrnuty v tabulce 2.2.

Tab. 2.2 Tabulka shrnující parametry jednodeskového počítače FEZ Cerbuino BEE, převzaná ze stránek výrobce (GHI Electronics LLC, 2013)

Procesor	168 Mhz 32-bit Cortex-M4
Hardwarový systém jádra	Cerberus Chipset
Systémová platforma	.NET Micro Framework
Uživatelsky přístupná FLASH paměť	348 KB
Uživatelsky přístupná paměť RAM	119 KB Non Ethernet 104 KB Ethernet
GPIO	34
PWM	14
Analogové vstupy	10
UART	2
Podpora SPI, I^2C	Ano
Podpora síťových protokolů	Ethernet TCP/IP, WiFi, SSL
CAN	Ano
Analogové výstupy	2
USB klient	Ano
USB host	Ano
1-Wire	Ano
Hodiny reálného času	Ano
Paměťové karty	Ano
Rozsah provozní teploty	0 až +70°C
Rozměry	55,9 x 81,3 x 13,9 mm
Hmotnost	28 g
Velikost RAM	8 KB Non-Ethernet 12 KB Ethernet
Počet .NET Gadgeteer slotů	3
Typy .NET Gadgeteer slotů	2 x Y 1 x O 2 x A 2 x P 1 x I 1 x S 1 x K 2 x U

3 LEGO Mindstorms Education NXT 2.0, EV3

Stavebnice LEGO Minstorms Education sestává z řídicí jednotky a jednotlivých periferií. Verze NXT 2.0 je na trhu dostupná ke koupi od roku 2009, kdy původní stavebnice sestávala z 619 součástí (Hurbain, 2000). V roce 2013 byla vydána verze nová, jejíž hlavní předností je podpora vzdáleného ovládání pomocí chytrého telefonu a byla označena verzí EV3.

3.1 Počítač Intelligent Brick

Řídicím prvkem je Intelligent Brick, jenž zastává roli řídicího programovatelného počítače.

Obr. 3.1 Fotografie řídicí jednotky NXT 2.0 Intelligent Brick (Hurbain, 2000)

Obr. 3.2 Fotografie řídicí jednotky EV3 Intelligent Brick (The LEGO Group, 2013b)

Parametry jednotlivých řídicích jednotek jsou uvedeny v tabulce 3.1 (The LEGO Group, 2013a).

Tab. 3.1 Tabulka s parametry řídicích počítačů stavebnice LEGO Mindstorms NXT 2.0 a LEGO Mindstorms EV3

	NXT 2.0 Intelligent Brick	**EV3 Intelligent Brick**
Procesor	ARM 7 48 MHz Atmel AT91SAM7S256, 256 KB flash paměť, 64 KB RAM	ARM 9 300 MHz TI Sitara AM1808 16 MB flash paměť 64 MB RAM OS Linux
Zobrazovací jednotka	100 x 64 LCD černobílý	178 x 128 LCD černobílý
Vstupy	4 x RJ12 pro senzory	4 x RJ12 pro senzory
Výstupy	3 x RJ12 pro motory	4 x RJ12 pro motory
USB	Ano	Ano
USB Host	Ne	Ano
Internet (USB Host)	Ne	Ano
Bluetooth	Ano	Ano
WiFi	Ne	Ano
Audio výstup	Reproduktor	Reproduktor
Tlačítka	4 x tlačítko pro spuštění/programování programu	3 x podsvícené tlačítko pro spuštění/programování programu

3.2 Periferie

Do řídicí jednotky jsou prostřednictvím 6-ti PINových konektorů RJ12 zapojeny jednotlivé periferie. Prostřednictvím těchto konektorů lze komunikovat prostřednictvím protokolu I^2C (OpenElectrons, 2012).

Stavebnice je navržena s ohledem možnost připojení dvou druhů periferií:

- Senzory
- Aktuátory

Senzory slouží pro snímání informací z okolí a vyhodnocené signály vysílají řídicí jednotce prostřednictvím portu, do kterého jsou zapojeny.

Aktuátory jsou akční členy stavebnice. Ve zmíněné stavebnici je pouze jeden druh aktuátoru a tím je servomotor.

Na webových stránkách společnosti LEGO lze zakoupit mnohá zařízení pro LEGO Mindstorms Education (The LEGO Group, 2015). Níže uvedené periferie s podrobnostmi jsou převzaty z návodu pro stavebnici LEGO Mindstorms Education EV3 (The LEGO Group, 2013b).

Obr. 3.3 Označení jednotlivých PINů konektoru RJ12 (vlevo) a porty 1-4 na logické kostce EV3 (vpravo)

3.2.1 Motory

V základní sadě LEGO Mindstorms Education EV3 jsou k dispozici dva servomotory rozdílných vlastností a jsou označovány jako velký a střední.

Oba motory lze ovládat s rozlišením 1° v rozsahu 360° v libovolném smyslu chodu. Mají vestavěný senzor natočení a lze provádět řízení natáčení s přesností až jednoho stupně. U středního servomotoru je senzor natočení poměrně menší a lehčí. Díky menším setrvačným silám je tedy při stejných otáčkách odezva středního servomotoru rychlejší.

Tab. 3.2 Srovnání kinematických a mechanických vlastností servomotorů

		Velký	**Střední**
Otáčky za minutu	[s^{-1}]	160 - 170	240 - 250
Kroutící moment za chodu	[Ncm]	20	8
Kroutící moment za klidu	[Ncm]	40	12

Obr. 3.4 Velký servomotor (vlevo) a střední servomotor (vpravo)

3.2.2 Senzory

Senzor barvy

Jedná se o digitální senzor měřící intenzitu světla dopadající na čidlo umístěné na čelní straně senzoru. Při měření úrovně intenzity dopadajícího světla je důležité dbát na natočení čidla pro zachování požadované citlivosti. Vzorkovací frekvence senzoru je 1 kHz a výstupní informací je světlo o jedné ze sedmi podporovaných barev nebo hodnota intenzity světla v rozsahu hodnot 0 - 100 (velmi malá intenzita - velmi velká intenzita) dopadající na čidlo. Barevný senzor tedy umožňuje provoz ve třech různých variantách: snímání barev, snímání odražené světelné intenzity a snímání okolní intenzity světla.

Obr. 3.5 Senzor barvy stavebnice LEGO MIndstorms Education EV3

Gyroskopický senzor

Gyroskopický senzor slouží dle návodu pro snímání rotačního pohybu kolem jedné osy v libovolném smyslu otáčení. Jeho výstupem je informace o počtu stupňů natočení za jednu vteřinu, kdy maximální snímaná rychlost natáčení je 440° za vteřinu s přesností ±3° na čtvrtinu jedné otáčky senzoru.

Obr. 3.6 Gyroskopický senzor

Senzor dotyku

Jedná se o pasivní analogový senzor, jehož výstupem je nelineární charakteristika míry zamáčknutí červeného tlačítka.

Obr. 3.7 Senzor dotyku

Ultrazvukový senzor vzdálenosti

Tento digitální senzor je schopen měřit vzájemnou vzdálenost od objektu umístěného ve směru nastavení čidel umístěného maximálně 250 cm daleko s přesností ± 1 cm. Senzor lze provozovat ve dvou snímacích variantách: přítomnostní a měřící. O momentálně nastavené variantě snímání indikuje dioda umístěná nad čidly senzoru. Pokud dioda kontinuálně svítí, znamená to, že senzor snímá v měřícím režimu a jeho výstupem je vzdálenost od vhodně umístěných předmětů. Pokud dioda svítí střídavě, je aktivní přítomnostní režim a senzor je schopen detekovat jiné ultrazvukové senzory v měřícím režimu.

Obr. 3.8 Ultrazvukový senzor vzdálenosti

Senzor teploty

Digitální senzor teploty je schopen měřit teplotu v rozsahu od -20°C do 120°C s přesností ±0.1 °C. Snímaná hodnota teploty je předpokládaná na zazším konci čidla (kovová tyč).

Obr. 3.9 Senzor teploty

3.3 Způsoby programování

V základní sestavě LEGO Mindstorms Education lze využít dvou metod tvorby programu. Možné je stáhnout ze stránek společnosti LEGO vývojový software nebo tvořit programy prostřednictvím grafického prostředí logické kostky stavebnice. Obě tyto metody jsou převzaty z návodu stavebnice LEGO Mindstorms Education EV3 (The LEGO Group, 2013b).

3.3.1 LEGO Mindstorms Education EV3 BRICK PROGRAM

Jedná se o vývojové prostředí nahrané v paměti logické kostky LEGO Mindstorms Education a slouží pro tvorbu řídicích programů. Na následujícím obrázku je zobrazeno uživatelské rozhraní vývojového prostředí logické kostky.

Obr. 3.10 Vývojové prostředí logické kostky LEGO Mindstorms Education EV3 ve fázi tvorby programu

Po založení nového programu je v grafickém prostředí zobrazeno blokové schéma, kde jednotlivé bloky představují předprogramované funkce ovládající zapojené moduly. Nově založený program obsahuje pouze dva bloky, po jejichž spuštění není proveden žádný zásah ve smyslu zapojených modulů (Obr. 3.10 - vlevo).

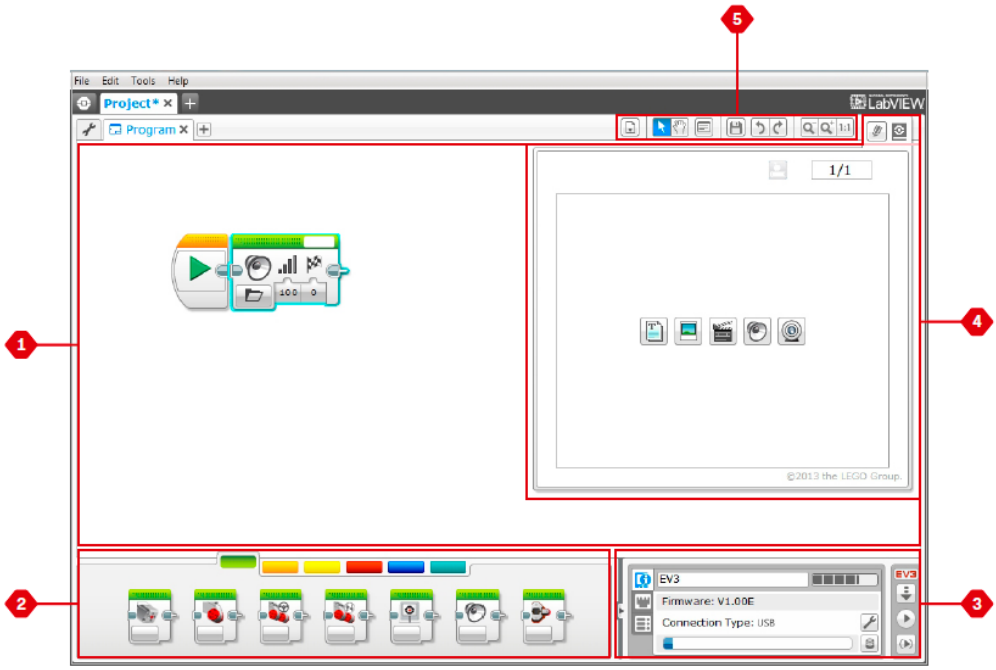
Základní funkční bloky jsou děleny do dvou kategorií "Action" a "Wait". Základní myšlenkou je seřazení jednotlivých Action bloků, které budou spuštěny po splnění podmínky předcházejících Wait bloků. V závislosti na vyznačených vazbách jsou po spuštění programu jednotlivé funkce vykonávány sekvenčně. Na obrázku 3.10 je vpravo zobrazen příklad programu, který po spuštění uvede do provozu servomotory zapojené do portů B a C. Z obrázku není zřejmé, jak dlouho a jakým smyslem se bude motor otáčet. Každý funkční blok má nastavitelné parametry, prostřednictvím kterých lze tyto vlastnosti provozu v rámci tvorby programu modifikovat.

3.3.2 LEGO Mindstorms Education EV3 software

Jedná se o vývojový software pro tvorbu programů LEGO Mindstorms určený pro osobní počítače (The LEGO Group, 2013b). Software je podporován operačními systémy Windows (XP, Vista, 7, 8) (32/64 bit) a Macintosh Mac (10.6, 10.7, 10.8). Vývojové prostředí je založeno na software společnosti National Instruments LabVIEW. Mezi hlavní přednosti dle návodu výrobce patří:

- Grafické a intuitivní uvedení do problematiky programování
- Obsáhlé a užitečné nástroje pro záznam dat
- Digitální sešit s navrhovanými instrukcemi a záznamy minulých zjištění
- Možnost přirozeně se inspirovat a získat zájem o témata v otázkách vědy, technologií, strojírenství a matematiky.

Po založení nového programu je ve vývojovém prostředí automaticky založen nový projekt, pod který následně spadají jednotlivé programy, experimenty, obrázky, zvuky, videa a instrukce. Všechny tyto objekty jsou přístupné prostřednictvím otevření složky projektu. Na následujícím obrázku je vývojové prostředí LEGO Mindstorms Education EV3 software, kdy projekt obsahuje jeden program, který po spuštění přehrává zvuk prostřednictvím reproduktoru logické kostky.



Obr. 3.11 Vývojové prostředí software LEGO Mindstorms Education EV3 pro osobní počítače ve fázi tvorby programu

Na obrázku 3.11 je vyznačeno pět nástrojů užívaných při tvorbě programu ve vývojovém prostředí software LEGO Mindstorms Education EV3:

1. Programové schéma - Princip tvorby blokových schémat je obdobný jako u tvorby programu prostřednictvím vývojového prostředí logické kostky.
2. Paleta bloků - Prostřednictvím této palety lze přistupovat ke všem podporovaným blokům vývojového software. Bloků je k dispozici podstatně více, než je tomu u vývojového prostředí logické kostky a jejich význam je následně uveden.
3. Panel hardware - V panelu hardware jsou zobrazeny jednotlivé parametry logické kostky LEGO Mindstorms připojené k osobnímu počítači.
4. Editor obsahu - Editor obsahu je určen pro vedení záznamů o tvorbě vlastního programu. Programátor má díky tomuto nástroji možnost své projekty v libovolných fázích rozšířit o poznámky ve formě textu, audio-video záznamů, obrázků a zvukových záznamů.
5. Menu programu - Uvedené menu obsahuje nástroje určené pro správu vlastního programu.

Významným rozdílem mezi software pro osobní počítače a vývojovým prostředím, jímž disponuje Logic Brick, je velikost, kterou jednotlivé software zabírají v programové paměti. Paměťová zařízení obsažená ve stolních počítačích disponují kapacitami v řádech stovek GB až několika TB a z tohoto důvodu je v software pro osobní počítače obsaženo více předprogramovaných funkcí ve formě funkčních bloků, zvuků a dalších.

Akční bloky se tedy nedělí pouze do dvou skupin, jak je tomu u vývojového prostředí Logic Brick. Každý akční blok vývojového prostředí pro stolní počítače spadá do jedné ze šesti skupin definovaných výrobcem a skupiny jsou navzájem odlišeny prostřednictvím barevného značení do jednotlivých kategorií:

Akční bloky (zelená)

Funkcí akčních bloků je provoz motorů, překreslení displeje Logic Cube, reprodukce zvuku nebo indikace prostřednictvím diod Logic Brick dle stanovených parametrů.

Provozní bloky (oranžová)

Provozní bloky jsou obdobou pro podmíněné spouštění bloků, se kterými jsou v přímé vazbě. Při spojení s tímto blokem lze uvést jiný blok do stavu spustit jednorázově, čekat, spustit opakovaně, volitelné spuštění dle hodnot stanovených proměnných a opakované spuštění s možností přerušení.

Bloky senzorů (žlutá)

Výstupem těchto bloků je vždy informace přijatá prostřednictvím připojené periferie, tlačítka Logic Brick nebo stavu časovače. Pomocí těchto bloků lze přistupovat k jednotlivým stavům a naměřeným hodnotám připojených periferií a dosáhnout tak řízení za chodu programu.

Datové bloky (červená)

Datové bloky umožňují použití matematických funkcí a definici vlastních proměnných a konstant. Prostřednictvím těchto bloků lze užít matematické operace s prvky pole číselných nebo logických hodnot, zaokrouhlení, porovnání, rozsah, převod na text a náhodné generování čísel.

Bloky pokročilých funkcí (modrá)

Pokročilé funkce umožňují přístup k datům v paměti Logic Brick, záznam měřených dat, komunikaci s jiným zařízením prostřednictvím zpráv, správu připojení Bluetooth, přístup k nezpracovaným hodnotám zapojených senzorů, vypnutí regulace chodu motoru, přepnutí smyslu chodu motoru a vynucené zastavení chodu programu.

Uživatelské bloky (světle modrá)

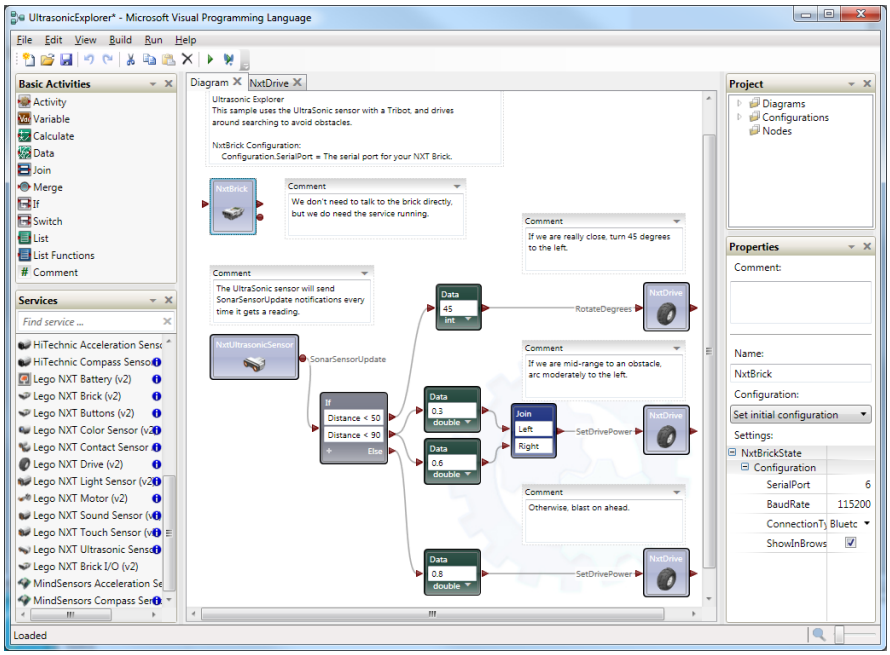
Tato skupina slouží pro návrh uživatelských bloků, které mohou slučovat funkce bloků uvedených v předchozích kategoriích.

Metody uvedené v kapitolách 3.3.1. a 3.3.2. pro programování Logic Brick stavebnice LEGO Mindstorms Education, jsou k dispozici v základní sestavě a tvorba programu je omezena na možnosti jednotlivých funkčních bloků stanovených výrobcem. Následně jsou uvedeny vývojové nástroje, které umožňují tvorbu programu ve formě programovacího jazyku.

3.3.3 Microsoft Robotics Developer Studio 4

Jedná se o vývojové prostředí společnosti Microsoft, určené pro operační systém Windows, a které je založeno na platforme .NET Framework. Všechny informace o software jsou převzaty ze stránek výrobce (Microsoft Corporation, 2012). Prostřednictvím tohoto software lze vyvíjet programy pro mnohá zařízení, včetně počítače Logic Brick stavebnice LEGO Mindstroms Education.

K tvorbě programu prostřednictvím Microsoft Robotics Developer Studia lze přistupovat několika metodami. Programátor nemusí napsat žádnou část kódu, kdy lze využít nástroje Visual Programming Language ve formě spojování vhodných služeb a aktivit pouze přetahováním ze seznamu do programového blokového schématu (Kelly, 2010).



Obr. 3.12 Nástroj Microsoft Visual Programming Language vývojovéhoprostředí Microsoft Robotics Developer Studio (Kelly, 2010)

Vývojové prostředí Microsoft Visual Programming Language vypadá obdobně, jak je tomu u software LEGO Mindstorms Education EV3 pro osobní počítače. Možnosti nastavení jednotlivých bloků jsou u nástroje společnosti Microsoft značně rozsáhlejší a jednotlivé bloky i aktivity lze programovat v jazyce C#.

3.3.4 NeXT Byte Codes (NBC)

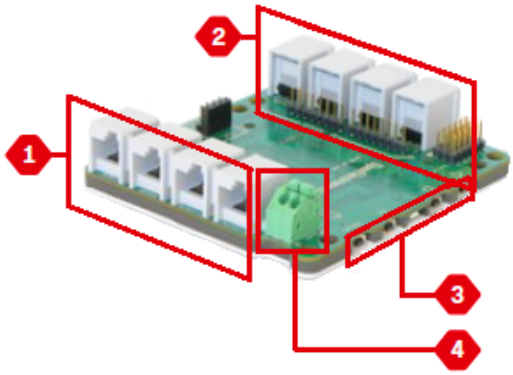
Nejedná se o vývojové prostředí, avšak o programovací jazyk, určený k tvorbě software pro LEGO Mindstorms Education NXT. Podrobné informace jsou uvedeny v návodu (Hansen, 2011). Pro tento programovací jazyk není definováno jednotné vývojové prostředí. Hlavní je, aby měl projekt na začátku zdrojového kódu umístěn odkaz na hlavičkový soubor obsahující funkce pro kompilaci NBC funkcí a kompilátor při své funkci nehavaroval. Tohoto je dosaženo prostřednictvím preprocesoru, který je součástí NBC a zprostředkovává přepis programového kódu z jazyka NBC do jazyka C před vlastní kompilací.

Tato metoda tvorby programu neuvažuje vlastní nahrání překompilovaného strojového kódu do paměti LEGO Mindstorms Logic Brick a proto je nutné využít nástroje schopné této funkce.

Vzhledem k vlastnostem výše uvedených metod programování řídicí jednotky Logic Brick stavebnice LEGO Mindstorms Education lze vyvodit, že lze i s mizivými znalostmi teorie programování vytvořit jednoduché funkční programy. Mezi nejprimitivnější metody se řadí tvorba programu ve formě blokových diagramů. Takto vytvořená schémata jsou značně přehlednější, než program napsaný v programovacím jazyce, avšak funkce každého bloku jsou omezeny pouze na jeho vlastní možnosti nastavení a okolní vazby. Pomocí metod, kdy je projekt tvořen prostřednictvím programovacího jazyka může vývojář využít všechny funkce logické kostky i jednotlivých periferií, kdy je omezen pouze konstrukcí jednotlivých elektrických zařízení. Jako nejvhodnější vývojový software bych označil Microsoft Robotics Developer Studio, který umožňuje tvorbu software libovolnou ze zmiňovaných metod, aniž by bylo nutné zakládat nový projekt.

4 Vývojová deska NXShield-Dx

Pro řešenou úlohu jsou na stránkách výrobce vývojové desky (OpenElectrons, 2010) k dispozici dva důležité návody pro stáhnutí. První návod s názvem "NXShield-User-Guide.pdf" (OpenElectrons, 2011) je určen k obecnému seznámení s vývojovou deskou a možnostmi jejího užití. Pro podrobnější seznámení s vývojovou deskou slouží návod s názvem "NXShield-Advanced-Developer-Guide.pdf", ve kterém jsou obsaženy bližší specifikace ohledně ovládání a doporučených metod používání. Princip přístupu k jednotlivým senzorům a motorům zapojených do této vývojové desky spočívá ve čtení a přepisování hodnot registrů NXShield. Prostřednictvím modifikace jejích hodnot lze nastavovat řídicí parametry, číst data naměřená senzory a nebo také komunikovat se zapojenými zařízeními, které podporují komunikaci I^2C. Dále jsou uvedeny převzaté informace ze zmíněných návodů, které jsou důležité pro řešenou problematiku ovládání senzorů.



Obr. 4.1 Fotografie vývojové desky NXShield-Dx s vyznačenými funkčními prvky

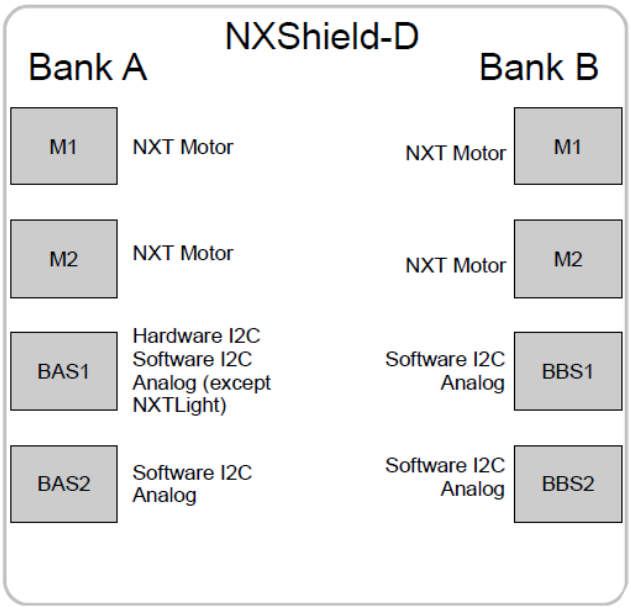
Na obrázku 4.1 jsou na desce NXShield-Dx označena osazená rozhraní, která jsou v úloze užívána. Vysvětlení označených prvků je dále uvedeno v rámci rozboru jednotlivých témat.

Možnosti napájení

Pro požadované napájení vývojové desky je nutné použít zeleného terminálu (Obr. 4.1 - 4), na který je povoleno přivést maximálně 10,5 V stejnosměrné napětí. NXShield-Dx nelze napájet prostřednictvím žádného z pinů osazených Arduino sběrnic. Pro chod zapojených servomotorů je vyžadováno napájecí napětí minimálně 6,6 V.

Specifikace portů

Jednotlivé porty pro připojení LEGO Mindstorms motorů a senzorů jsou na obrázku 4.1 označeny oblastmi 1 a 2. Požadované funkce zapojených zařízení lze dosáhnout pouze za předpokladu, že zvolený port je určen pro daný typ periferie. Dovolené možnosti zapojení periferií jsou uvedeny na obrázku 4.2.



Obr. 4.2 Znázornění jednotlivých portů desky NXShield řady D s možnostmi zapojení jednotlivých druhů periferií

Každý port má přiděleno několik registrů se sedmi-bitovou adresou pro možnost komunikace přes Arduino sběrnice prostřednictvím protokolu I^2C. Pokud je tedy zapojen například senzor do portu BAS2, je nutné komunikační signály vést tak, aby bylo možné navázat spojení se slave zařízením, sběrnicí Bank A, a pak lze přistupovat k vlastním registrům BAS2.

4.1 I^2C adresace slave zařízení

V následujících tabulkách jsou uvedeny jednotlivé adresy registrů s vysvětlením jejich možností užití při čtení a přepisování obsažených hodnot.

Na obrázku 4.2 jsou zobrazeny dvě sběrnice, kterými je osazena vývojová deska. Uvedené porty na sběrnicích slouží pro přenos signálů mezi registry desky a zapojenými zařízeními. Z principu protokolu I^2C vyplývá (NXP Semiconductors, 2014), že každé zařízení na komunikačním kanále musí mít unikátní adresu, stejně jako jednotlivé registry tohoto zařízení. Pro přístup k registrům sběrnic Bank A a Bank B jsou uvedeny jejich adresy v tab. 4.1.

Tab. 4.1 Tabulka s 7-bit adresami pro přístup k registrům jednotlivých sběrnic vývojové desky NXShield

Čtení registru	Zápis registru	Adresa registru
Bank A		0x03
Bank B		0x04

Obě zmíněné sběrnice Bank A a Bank B sestávají z portů, kdy vždy dva na sběrnici jsou určeny pro zapojení servomotoru a dva pro zapojení senzoru. Jediným rozdílem mezi porty obou sběrnic se shodnou adresou je port BAS1 sběrnice Bank A, který na úkor ztráty podpory světelného senzoru stavebnice LEGO Mindstorms podporuje komunikaci na bázi hardwarového I^2C protokolu. Po navázání spojení prostřednictvím jedné ze dvou sběrnic lze přistupovat ke všem registrům pod ní spadajících portů. Následující uvedené informace ohledně adresace jsou vztaženy vzhledem k jedné ze sběrnic vývojové desky. Pro vlastní navázání komunikace se slave zařízením, jednou ze sběrnic, je potřeba použít I^2C signálů pro aspoň jeden port na sběrnici dle tabulky 4.7. Prostřednictvím sběrnice lze dále přistupovat k registrům dalších portů na sběrnici, nikoli však k registrům portů na sběrnici druhé. Za předpokladu zapojení jednoho signálu SCL a SDA je tedy možné využívat zařízení zapojené do jedné sběrnice, maximálně dva motory a dva senzory.

Následující tabulky 4.2 až 4.5 jsou věnovány nastavování a čtení řídicích parametrů pro zapojené servomotory. Vývojová deska NXShield-Dx je schopna dle nastavení příslušných hodnot registrů řídit zapojený servomotor s ohledem na: rychlost otáčení, polohu natočení a dobu otáčení. Každý motor, respektive port určený pro zapojení motoru, má přidělen vlastní adresní prostor, kdy každý jednotlivý byte má význam pro nastavení řídicích parametrů motoru.

Tab. 4.2 Tabulka s adresami pro ovládání servomotorů zapojených do portu Motor 1 nebo Motor 2

Čtení registru pro motory	**Zápis registru pro motory**		**Adresa registru pro port**	
			Motor 1	**Motor 2**
Nastavení enkodéru.	Nastavení enkodéru.		0x42	0x4A
Čtení rychlosti otáčení	Nastavení rychlosti otáčení		0x46	0x4E
Čas točení motoru	Nastavení času otáčení motoru		0x47	0x4F
Příkazy pro nastavení režimu řízení motoru (Tab 4.4)	Příkazy pro nastavení režimu řízení motoru (Tab 4.4)	B	0x48	0x50
		A	0x49	0x51
Natočení enkodéru			0x62	0x66
Stav motoru (Tab. 4.3)			0x72	0x73

Uvedení motoru do chodu lze provést prostřednictvím nastavení hodnot dle požadovaného způsobu řízení a poté do registru pro nastavení režimu řízení zapsat hodnotu byte, která má MSB roven 1. Pokud má MSB na adrese 0x49 pro port Motor-1 nebo 0x51 pro port Motor-2 hodnotu 1, jsou vývojovou deskou načteny hodnoty nastavení enkodéru, rychlosti otáčení, času pro otáčení, parametrů řízení a motor zapojený v příslušném portu je za předpokladu požadovaného napájení uveden do provozu.

Při přepisování hodnot registrů je nutné brát ohled na nastavený režim řízení servomotoru. Právě nastavený režim řízení pro uvažovaný servomotor je možné přečíst z registru na příslušné adrese uvedené v tabulce 4.2 pod názvem "Stav motoru". Velikost

přečteného registru je jeden byte a každý jednotlivý bit nese informaci dle principu uvedeného v tabulce 4.3. Lze také vyhodnotit důvod nežádoucího zastavení motoru.

Pokud je potřeba změnit režim řízení servomotoru, je nutné přepsat hodnotu registru pro uvažovaný servomotor na adrese uvedené v tabulce 4.2 pod názvem "Příkazy pro nastavení režimu řízení motoru". Hodnota registru na uvažované adrese má velikost jeden byte, kdy změna hodnoty každého jednotlivého bitu má za následek přenastavení způsobu řízení servomotoru. Význam hodnot jednotlivých bitů, prostřednictvím kterých NXShield-Dx modifikuje způsob řízení servomotoru, je uveden v tabulce 4.4. Motor je prostřednictvím nastavení tohoto registru uveden do provozu ve chvíli, kdy je MSB nastaven na hodnotu 1.

Tab. 4.3 Tabulka s vysvětlením významů bitů představujících aktuální stav jednotlivých motorů

Pokud hodnota bitu = 1	**Pořadí bitu**
Řízení rychlosti je zapnuto. NXShield řídí otáčení motoru vzhledem k nastavené rychlosti otáčení.	LSB Bit 0
Rychlost otáčení motoru se mění plynule.	Bit 1
Motor je napájen (zároveň se nemusí natáčet).	Bit 2
Hlídání natočení je zapnuto. Pokud se motor nenatáčí na nastavenou pozici tak je zabrzděn na stanovené pozici.	Bit 3
Brzdění motoru je zapnuto.	Bit 4
Přetížení motoru., kdy vnější síla brání motoru dosáhnout nastavené rychlosti otáčení.	Bit 5
Režim spouštění motoru na definovanou dobu.	Bit 6
Přetížení motoru., kdy vnější síla způsobí zastavení otáčení motoru.	MSB Bit 7

Tab. 4.4 Tabulka s vysvětlením významů bitů představujících režim provozu jednotlivých motorů

Pokud hodnota bitu nastavena na hodnotu = 1	**Pořadí bitu**
Řízení rychlosti otáčení je zapnuto.	LSB Bit 0
Změna rychlosti otáčení je řízena s ohledem na plynulost.	Bit 1
Relativní řízení natočení.	Bit 2
Absolutní řízení natočení.	Bit 3
Brzda při zastavení otáčení je zapnuta.	Bit 4
Hlídání nastavené pozice natočení.	Bit 5
Možnost natáčet motor po stanovenou dobu.	Bit 6
Potvrzení nastavení bitů a uvedení motoru do chodu.	MSB Bit 7

Pro každý motor jsou v tabulce 4.2 uvedeny dvě adresy pro nastavení režimu řízení servomotoru, adresa registru nastavení A a adresa registru nastavení B. Význam registru B je dle návodu rezervou pro budoucí funkce a hodnoty jeho jednotlivých bitů musí být nastaveny na 0. Vlastní nastavení režimu řízení a uvedení servomotoru do chodu lze tedy provést pouze prostřednictvím přenastavení hodnot zmíněných registrů nastavení A.

V tabulce 4.5 jsou uvedeny hodnoty příkazů pro okamžité zastavení chodu servomotoru v hexadecimálním tvaru, prostřednictvím kterých lze zastavit otáčení motoru a zároveň uvést nastavené parametry metod řízení motorů do původního stavu nebo také zapnout/vypnout zabrzdění otáčení při klidovém stavu motoru. Hodnotu pro příslušný příkaz je nutné zapsat do registru s adresou 0x41 prostřednictvím slave zařízení uvažované sběrnice.

Tab. 4.5 Tabulka s příkazy pro nastavení parametrů zastavení motorů

Příkaz nastavující zastavení motoru při parametrech	**Hodnota příkazu pro zapsání na adresu 0x41**	
	Motor 1	**Motor 2**
Nastavení hodnot enkodéru a parametrů provozu motoru na původního stav.	0x52	
Synchronizovaný příkaz motorům.	0x53	
Brzda je vypnuta.	0x61	0x62
Brzda je vypnuta pro oba motory.	0x63	
Brzda je zapnuta.	0x41	0x42
Brzda je zapnuta pro oba motory.	0x43	
Nastavení hodnoty enkodéru na původní hodnotu.	0x72	0x73

Pokud není specifikován režim řízení motoru prostřednictvím polohy enkodéru nebo stanovením požadované doby provozu, tak po uvedení do chodu běží motor neomezenou dobu. Ve zmíněném případě NXShield řídí pouze rychlost otáčení motoru. Výše byly v tabulkách uvedeny možnosti nastavení hodnot registrů týkajících se parametrů řízení zapojených servomotorů vývojovou deskou. Dále jsou popsány metody přístupu k zapojeným senzorům.

Tab. 4.6 Tabulka s adresami registrů portů určených pro senzory

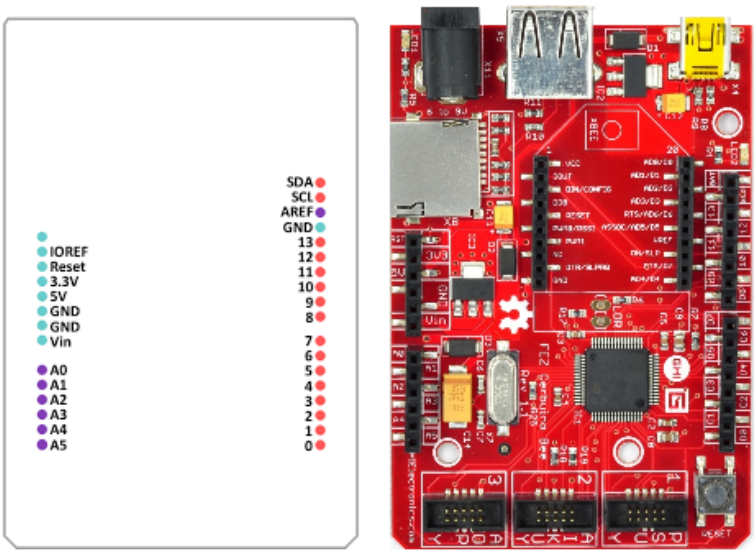
Čtení z registru pro senzory	**Zápis do registru pro senzory**	**Adresa registru pro port**	
		Senzor 1	**Senzor 2**
Čtení nastaveného režimu snímání	Nastavení režimu snímání	0x8A	0x8B
Čtení hodnoty senzoru LSB		0x8C	0x8E
Čtení hodnoty senzoru MSB		0x8D	0x8F

V tabulce 4.6 jsou uvedeny adresy registrů určených pro uložení naměřených hodnot zapojenými senzory a také adresy registrů s nastavením režimu snímání jednotlivých senzorů. Pokud je senzor pasivní a nevyžaduje napájení prostřednictvím portu, hodnotu registru je nutné nastavit 0x01. Aktivní senzory jsou napájeny 9V a hodnota registru pro nastavení režimu snímání musí být přepsána na 0x02.

4.2 Význam pinů Arduino headers pro přenos I²C signálů

Vývojová deska NXShield-Dx je navržena tak, aby plnila funkci řídicí jednotky pro zapojené periferie. Do vývojové desky jsou tedy zapojeny jednotlivé motory a senzory, kdy při vhodném nastavení registrů je dle kapitoly s adresacemi dosaženo požadovaného řízení motorů i čtení informací ze senzorů. Do samostatné vývojové desky nelze nahrát software, jak tomu bylo u logické kostky stavebnice LEGO Mindstorms. Za tímto účelem je nutné použít programovatelné zařízení, které je schopno komunikace s vývojovou deskou, tedy čtení a přepisování hodnot registrů NXShield-Dx. Vlastní komunikace je dle návodu uvažována prostřednictvím protokolu I²C. Jednodeskový počítač FEZ-Cerbuino-Bee je vhodným zařízením pro plnění zmíněných funkcí. V této podkapitole jsou rozebrány specifikace sběrnice předpokládané pro komunikaci mezi jednodeskovým počítačem FEZ-Cerbuino-Bee a vývojovou deskou NXShield-Dx.

Obě zmíněná zařízení jsou osazena Arduino kompatibilními sběrnicemi, prostřednictvím kterých je možné zprostředkovávat komunikaci pomocí I²C protokolu. Rozvržení jednotlivých pinů Arduino sběrnic u FEZ Cerbuino Bee je shodné s jednodeskovým počítačem Arduino Uno, který je dle výrobce zcela kompatibilní pro komunikaci s NXShield-Dx.



Obr. 4.3 Značení jednotlivých pinů Arduino kompatibilních sběrnic osazených na jednodeskovém počítači FEZ Cerbuino Bee

Tab. 4.7 Tabulka s označením jednotlivých pinů Arduino sběrnic počítače FEZ Cerbuino Bee pro komunikaci s se slave zařízeními NXShield-Dx

Signál	Označení pinu	Označení pinu	Signál
		AREF	-
		GND	-
		13	-
		12	LED_MODRA TLACITKO_VPRAVO
-	RESET	11	-
-	3.3V	10	-
-	5V	9	-
-	GND	8	LED_CERVENA TLACITKO_VLEVO
-	GND	7	SCL_BBS2
-	Vin	6	-
SDA_BAS2	A0	5	-
SDA_BBS1	A1	4	SCL_BBS1
SDA_BBS2	A2	3	-
LED_ZELENA TLACITKO_GO	A3	2	SCL_BAS2
SDA_BAS1	A4	1	-
SCL_BAS1	A5	0	-

Pro komunikaci s registry vývojové desky NXShield-Dx nejsou využívány všechny piny Arduino sběrnic. Funkční piny pro uvažovanou komunikaci s registry mají v tabulce 4.7 ve sloupci *Signál* uveden název přenášeného signálu nebo prvek, ke kterému je prostřednictvím pinu přistupováno. Signály s předponou SDA a SCL jsou základem pro komunikaci prostřednictvím protokolu I^2C, kdy SDA značí "synchronous data" a SCL "synchronous clock". Jednotlivé přípony u zmíněných signálů jsou pro definování specifického portu s příslušnými registry. Ostatní piny slouží pro přenos digitálních signálů pro čtení stavů zamáčknutí tlačítek a nastavení stavu zapnutí světelných diod.

V rámci tvorby software prostřednictvím platformy .NET Gadgeteer je možné zprostředkovávat I^2C komunikaci hardwarově nebo softwarově. Rozdílem mezi metodami je, že při softwarové komunikaci je SDA a SCL generován programem na libovolném GPIO pinu. Analogové a digitální Piny osazené Arduino sběrnice jsou GPIO a v uvažované aplikaci jsou využity pro softwarovou I^2C komunikaci dle významu uvedeném v tabulce 4.7.

Hardwarově generovaná komunikace má u použitého jednodeskového počítače výhodu menší výpočetní náročnosti (GHI Electronics LLC, 2015), protože je využit osazený logický obvod navržen pro generování příslušných SDA a SCL, kdy je potřebný

výpočetní čas ušetřen CPU. Při tvorbě software v platformě .NET Gadgeteer však není možné nastavit, na kterých pinech bude přítomen signál, protože hardwarová varianta protokolu je určena pro komunikaci s Gadgeteer moduly prostřednictvím gadgeteer slotů. HW protokol I^2C je tedy možné u jednodeskového počítače FEZ Cerbuino Bee použít pouze tehdy, když jsou signály na pinech gadgeteer slotů SCL a SDA vyvedeny na příslušné piny desky NShield-Dx.

4.3 Komunikace I^2C prostřednictvím pinů Arduino sběrnic

Jako vývojové prostředí je použit software společnosti Windows Visual Studio 2012, ve kterém musí být nainstalovány nástroje pro možnost tvorby software na platformě .NET Gadgeteer, jenž je postaven na platformě .NET Micro Framework. Značnou výhodou platformy .NET Gadgeteer pro řešenou problematiku tvorby programu na jednodeskový počítač FEZ Cerbuino Bee je podpora jeho procesoru kompilátorem a je tedy možné ihned po založení nového projektu software překompilovat a nahrát do paměti počítače prostřednictvím USB rozhraní.

Poslední problematika se týká naprogramování software pro vlastní generování a vyhodnocení přijatých informací komunikace I^2C. Jelikož je vývojová deska NXShield-Dx navržena pro připojení a komunikaci prostřednictvím Arduino sběrnic, je vhodné tuto variantu zapojení použít a naprogramovat komunikaci na softwarové bázi prostřednictvím pinů zmíněných sběrnic.

Balíček funkcí, obsahující také metody softwarového I^2C protokolu, je součástí .NET Micro Frameworku pod názvem GHI.OSHW.Hardware a je nutné ho do založeného gadgeteer projektu implementovat vložením kódu na začátek programu.

```
using GHI.OSHW.Hardware;
```

Nyní je možné kompilovat funkce softwarového I^2C. Nejprve je však nutné definovat hlavní parametry komunikace. Koncové slave zařízení, sběrnice Bank A nebo B, disponují svou unikátní adresou, nachází se na různých komunikačních kanálech a obsahují další vnitřní registry. Aby bylo možné přistupovat k registrům slave zařízení, je nutné založit objekt komunikačního rozhraní a prostřednictvím tohoto objektu založit objekt cílového zařízení pro komunikaci. Použitý kód pro založení objektu komunikačního rozhraní:

```
GHI.OSHW.Hardware.SoftwareI2CBus BAS1_bus =
new GHI.OSHW.Hardware.SoftwareI2CBus (SCL_PIN, SDA_PIN);
```

Pro založení objektu třídy SoftwareI2CBus jsou zapotřebí dva parametry datového typu Cpu.Pin, které jsou .NET Micro Framework směrníky na fyzické piny určené pro

přenos signálů. Pro komunikaci s registry portu BAS1 jsou použity piny FEZ Cerbuino Bee A5 pro SCL a A4 pro SDA. Piny je nutné v rámci parametrů funkce formulovat v příslušném formátu Cpu.Pin platformy .NET Micro Framework pro užívané zařízení. Je tedy nutné přidat referenci objektu GHI.Hardware.FEZCerb obsahující metody k provázání jednotlivých pinů Arduino sběrnic s piny procesoru, ke kterým mohou funkce .NET Micro Frameworku přistupovat. Poté již lze přetypovat slovní názvy jednotlivých pinů na datový typ Cpu.Pin.

```
Cpu.Pin A4 = GHI.Hardware.FEZCerb.Pin.PC1;
Cpu.Pin A5 = GHI.Hardware.FEZCerb.Pin.PA4;
```

Piny procesoru s vazbou na použité piny Arduino sběrnic jsou uvedeny v násedující tabulce 4.8 dle datasheetu výrobce (GHI Electronics, 2013a).

Tab. 4.8 Tabulka s názvy pinů procesoru počítače FEZ Cerbuino Bee, jejichž signály jsou vyvedeny na použité piny Arduino sběrnic

Pin Arduino sběrnice	Pin procesoru STM32F40X
D12	PB4
D8	PB13
D7	PC4
D4	PC15
D2	PB12
A0	PB1
A1	PA5
A2	PB0
A3	PC3
A4	PC1
A5	PA4

Pokud je založen komunikační kanál ve formě objektu softwarei2cBus, je možné na něm založit objekt zařízení, se kterým lze dále komunikovat. Za tímto účelem je v třídě softwarei2cBus implementována metoda Createi2cDevice, která na založeném objektu komunikačního rozhraní softwarei2cBus založí objekt zařízení. Vstupem pro metodu jsou dva argumenty, kdy prvním je sedmibitová adresa zařízení a druhým frekvence hodinového signálu SCL. Jako parametr sedmi-bitové adresy v hexadecimálním tvaru je pro příklad uvedena hodnota adresy sběrnice Bank A 0x03. Druhým parametrem je frekvence hodinového signálu SCL v jednotkách KHz, která je volena 100 KHz dle funkčních knihoven výrobce NXShield-Dx (OpenElectrons, 2014).

```
GHI.OSHW.Hardware.SoftwareI2CBus.I2CDevice BAS1_zarizeni =
BAS1_bus.CreateI2CDevice(0x03, 100);
```

Na založený objekt BAS1_zarizeni třídy softwarei2cDevice lze nyní aplikovat metody pro čtení a zápis hodnot registrů spadající pod fyzické zařízení. Nejprve je vhodné uvést princip jednotlivých funkcí a vysvětlit jak je v rámci I^2C zprostředkováno čtení a zápis informací na příslušnou paměťovou adresu.

Metoda pro zápis hodnoty vyžaduje posloupnost akcí, kdy počátkem zápisu je vyslání jednoho indikačního stavu start I^2C komunikace (SDA změna z HIGH na LOW, když SCL je LOW), následuje předání datových byte a ukončena je indikačním stavem stop (SDA LOW na HIGH, když SCL je LOW). Jak již bylo zmíněno, adresace je uvažována v podobě 7-bitových adres. Předávané pakety jsou tedy velikosti dvou a více byte, kdy pro zápis platí, že prvních 7 bitů prvního byte určuje cílovou adresu zařízení s registrem pro zápis, následující bit určuje, zda je požadavek o čtení nebo zápis, následující byte představuje adresu registru pro zápis na lince zařízení a poté následují jednotlivé byte s daty pro zapsání.

Na softwarově vytvořené zařízení je pro zápis možné aplikovat metodu Write(). Prvním argumentem metody je proměnná buffer datového typu byte[], která nese určitý počet byte pro zápis. Druhým argumentem je celočíselná hodnota, definovaná jako offset, prostřednictvím které je určena počáteční pozice pro čtení bitů bufferu, určeného pro zápis. Posledním argumentem je celočíselná hodnota označená jako count, pomocí které je určeno, kolik bytů z bufferu má být celkem zapsáno na adresu registru.

```
int Write(byte[] buffer, int offset, int count);
```

Nejprve je tedy dle principu I^2C nutné naplnit proměnné typu byte[] příslušnými hodnotami. První byte proměnné obsahuje od MSB 7-bitovou adresu zařízení , kdy LSB je 0 (sudá hodnota - zápis). Druhý byte obsahuje adresu registru zařízení pro zápis. Následující byte obsahuje vlastní hodnotu pro zápis. Kód pro zápis hodnoty 1 na registr s adresou 0x01, který je na lince vytvořeného objektu zařízení s adresou 0x03 vypadá tedy následovně:

```
byte[] byte_registr_data = new byte[2] { 0x01, 1 };
BAS1_zarizeni.Write(byte_registr_data,
int poc_pozice_zap = 0, int pocet_zapisovanych_byte = 2);
```

Metodu, s jakou bylo uvedeno zprostředkování zápisu hodnoty na adresu registru, je potřeba pro čtení hodnoty dále modifikovat. Pro čtení hodnoty registru na lince slave zařízení je potřeba nejdříve indikovat stavem Start, prvním byte volat 7-bitovou adresu zařízení s LSB rovno 0 (zápis), zapsat na adresu zařízení hodnotu adresy registru s daty pro čtení, poté zopakovat indikační stav start, opět zapsat adresu registru, nyní však s LSB 1 (čtení) a nyní lze na adrese zařízení číst požadovaná data registru. Po dokončení čtení je potřeba indikačního stavu Stop. Pro aplikaci uvedeného principu je vhodné zvolit metody WriteRead(), která umožňuje v jediném kroku zápis a čtení prostřednictvím objektu I^2C zařízení. Funkce se vstupními parametry je pro čtení vypadá následovně:

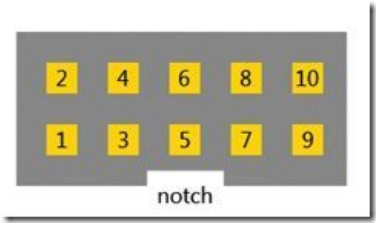
```
public bool WriteRead(byte[] writeBuffer, int writeOffset, int
writeLength, byte[] readBuffer, int readOffset, int readLength,
out int numWritten, out int numRead);
```

Výstupem metod Read(), Write() a parametry metody WriteRead() numWritten a numRead, jsou celočíselné hodnoty, představující počet úspěšně zapsaných nebo přečtených bytů. Počet byte, které byly neúspěšně zapsány nebo přečteny lze odvodit z rozdílu argumentů metod na třetích pozicích, definujících požadovaný počet byte pro zápis/čtení a hodnoty, kterou metoda navrací.

Uvedený princip komunikace prostřednictvím softwarově generovaných I^2C signálů prostřednictvím pinů Arduino kompatibilních sběrnic je metodou užitou výrobcem NXShield-Dx při užití podporovaných Arduino jednodeskových počítačů. Při zapojení počítače FEZ Cerbuino Bee přes Arduino kompatibilní sběrnice nebylo prostřednictvím uvedených metod dosaženo zprostředkování I^2C komunikace. Z tohoto důvodu bylo nutné přejít k jinému rozhraní přenosu komunikačních signálů, přenosu prostřednictvím pinů Gadgeteer slotu kompatibilním s typem I.

4.4 Komunikace I^2C prostřednictvím pinů Gadgeteer slotu typu I

Pro tuto metodu je na jednodeskovém počítači FEZ Cerbuino Bee k dispozici jeden Gadgeteer slot typu I, který disponuje dvěma piny, na kterých lze prostřednictvím platformy .NET Gadgeteer generovat signály SDA a SCL. Pro přenos signálů je nutné přemostit spojení mezi těmito dvěma piny Gadgeteer I kompatibilního slotu a příslušnými piny Arduino sběrnic vývojové desky NXShield-Dx. Pokud je s vývojovou deskou možné navázat pouze jedno I^2C komunikační rozhraní, lze komunikovat pouze s registry zařízení zapojenými do portů na jedné ze sběrnic NXShield-Dx. Sběrnice Bank A disponuje dvěma registry pro zapojení senzoru, které umožňují rozdílný přístup v rámci podporovaných senzorů a je vhodné ji zvolit jako slave zařízení. Rozhraní je zprostředkováno prostřednictvím pinů pro port BAS1 dle tabulky 4.7, signál SCL gadgeteer slotu je tedy vyveden do pinu vývojové desky NXShield-Dx, do kterého by byl zapojen pin A5 a signál SDA je vyveden do pinu na kterém by byl zapojen pin A4. Pro vlastní komunikaci je nutné také přemostit výstup uzemnění. Ze stránek výrobce zařízení (GHI Electronics, 2013b) využívající Gadgeteer rozhraní je převzato schéma slotu a níže uvedený význam pinů I kompatibilního slotu.



Obr. 4.4 Schéma gadgeteer socketu s číselným označením jendotlivých pinů

Tab. 4.9 Tabulka s významem jednotlivých pinů Gadgeteer I kompatibilního socketu dle číselného označení uvedeného na obrázku 4.4

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Socket I	+3.3V	+5V	GPIO!	[UN]	[UN]	GPIO	[UN]	SDA	SCL	GND

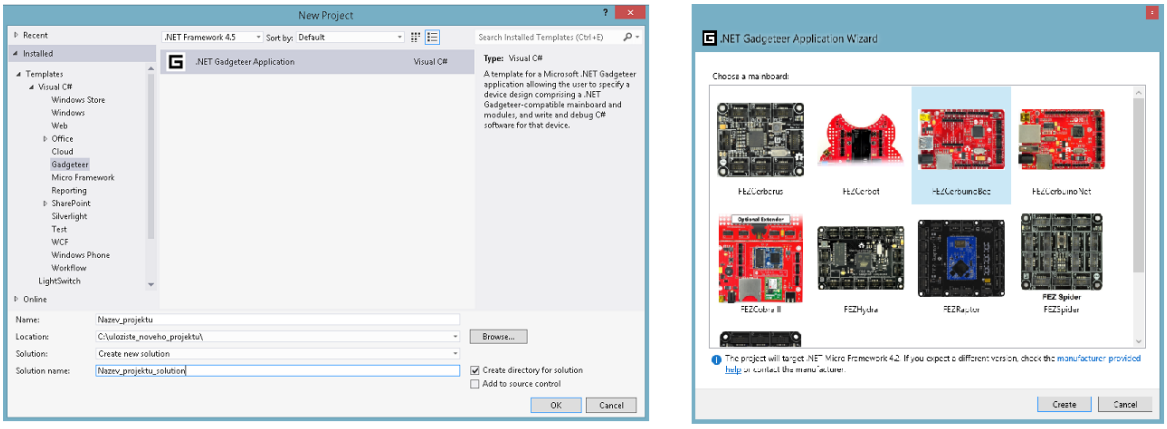
Prostřednictvím gadgeteer rozhraní byla úspěšně zprostředkována komunikace jednodeskového počítače s vývojovou deskou a užité metody jsou uvedeny v kapitole týkající se tvorby demonstrativních úloh.

5 Demonstrující úlohy na platformě .NET Micro Framework

Uvažované úlohy jsou napsány ve vývojové platformě .NET Gadgeteer, která je založena na platformě .NET Micro Framework. Prostřednictvím spojení výhod těchto platforem lze v uvažovaných úlohách využít moduly LEGO Mindstorms Education a zároveň moduly Gadgeteer. Základem pro vlastní úlohy bude senzor a velký motor stavebnice LEGO Minstorms, kdy vykreslení informací uživateli je zprostředkováno prostřednictvím Gadgeteer displeje. V předchozí kapitole byly uvedeny požadované metody přístupu k vývojové desce NXShield-Dx. Nyní je možné na základě uvedených principů komunikace navrhnout software, jenž je schopen plnit zvolené úlohy. Nejprve je však vhodné uvést způsob založení projektu a navrhnout programové funkce v jazyce C#, prostřednictvím kterých bude kód úloh přehlednější a úspornější. Prostřednictvím Gadgeteer displeje bude zprostředkováno uživatelské rozhraní a do počítače lze nahrát tedy software obsahující všechny navržené úlohy, kdy v rámci UI lze mezi jednotlivými programy přepínat pomocí tlačítek, jimiž je osazena vývojová deska NXShield-Dx. V rámci displeje je také zobrazována nápověda pro vhodné zapojení jednotlivých modulů stavebnice LEGO Mindstorms Education.

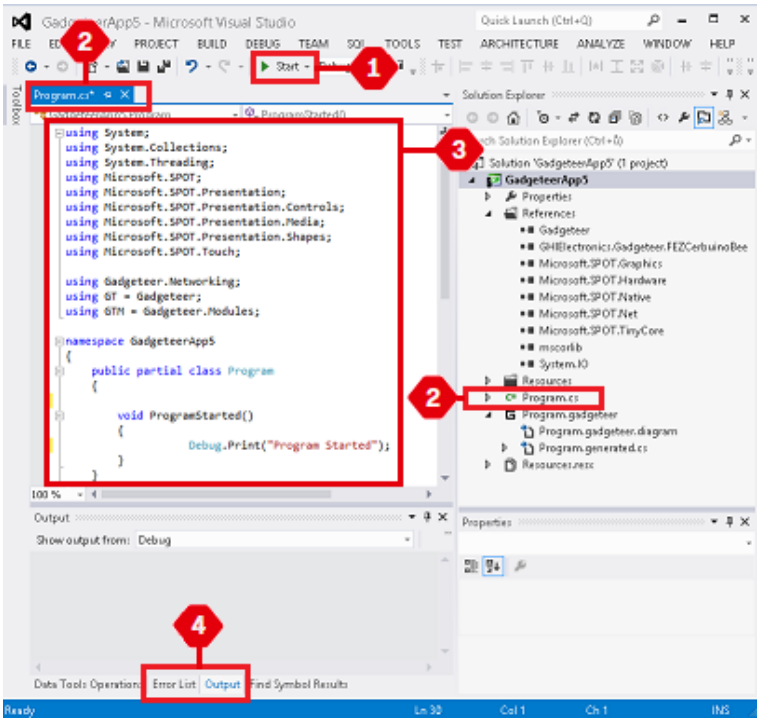
5.1 Založení projektu .NET Gadgeteer

Nejprve je ve vývojovém prostředí MS Visual Studio 2012 založen nový projekt C# .NET Gadgeteer pro platformu .NET Framework 4.5. Tento proces je zobrazen na obrázku 5.1, kde se vlevo nachází část zobrazení vývojového prostředí po volbě založení nového projektu. Po vyplnění příslušných polí a potvrzení tlačítkem OK je vygenerováno nové okno požadující zvolení použitého jednodeskového počítače (Obr. 4.4 - vpravo). Pokud je platforma .NET Micro Framework správně nainstalována, je možné zvolit použitý jednodeskový počítač FEZ Cerbuino Bee a zmáčknutím tlačítka Create založit nový projekt.



Obr. 5.1 Snímky oken vývojového prostředí MS Visual Studio 2012 při zakládání nového C# .NET Gadgeteer projektu - okno s výběrem typu projektu (vlevo), okno s výběrem použitého jednodeskového počítače (vpravo)

Pokud nedošlo při zakládání uvažovaného projektu k chybě, tak jsou vygenerovány příslušné soubory projektu s referencemi jmenných prostor .NET Micro Framework a .NET Gadgeteer. Hlavní část programu je obsažena ve zdrojovém souboru Program.cs (Obr. 5.2 - 2), k jehož hlavičce jsou při tvorbě nového projektu přidány příslušné reference s metodami .NET Micro Framework, potřebnými pro správnou funkci software na použitém hardware. Program (Obr. 5.2 - 3) je v této úvodní fázi kompilovatelný a lze jej nahrát do paměti počítače, kdy jeho jedinou základní funkcí je vypsat prostřednictvím sériové linky na příkazový řádek ladícího okna vývojového prostředí text "Program Started" po provedení inicializačních funkcí. Je vhodné zkusit přejít do ladícího režimu a nahrát tuto základní verzi programu do paměti počítače. Pokud je prostřednictvím výstupu ladícího prostředí indikován přijatý řetězec "Program Started", je to jasným důkazem založení projektu vhodného pro použitý počítač, tedy úspěšné kompilace a nahrání programu do paměti počítače s následným požadovaným chodem programu. Ladící režim je spuštěn po zmáčknutí tlačítka Start (Obr. 5.2 -1), což má za následek spuštění procesů kompilace, následné nahrání programu do paměti, restartování chodu jednodeskového počítače a zahájení komunikace prostřednictvím sériové linky po její vlastní inicializaci. Pokud dojde během libovolného ze zmíněných procesů k chybě, tak je indikována v panelu Error List a ladící prostředí není spuštěno. Panel pro přepnutí mezi zobrazováním chyb a výstupu sériové linky je označen na obrázku 5.2 jako 4.



Obr. 5.2 Vývojové prostředí MS Visual Studio 2012 ve stavu po založení nového projektu .NET Gadgeteer s označenými prvky probíranými v této kapitole

5.2 Definice programových funkcí

Funkce využité v uvažovaných úlohách jsou navrženy s ohledem na dostupné moduly a nepodporují veškerá dostupná zařízení stavebnice LEGO Mindstroms Education. Veškeré metody přístupu k užitým zařízením NXShield-Dx (zapojené senzory, motory a osazená tlačítka) vychází z principů uvedených v kapitole 4 o vývojové desce.

Inicializace

Pro schopnost přistupovat k jednotlivým registrům je nutné nejprve vytvořit objekty pro komunikaci I^2C. Proměnná i2c_gadgeteer_socket je objektem představující použitý gadgeteer socket označený číslem 2. Tento socket je typu I a jako jediný z osazených socketů na desce FEZ Cerbuino Bee podporuje přenos I^2C signálů SCL a SDA (GHI Electronics, 2015). Všechny uvažované moduly jsou zapojeny do sběrnice BankA, která má dle tabulky 4.1 adresu 0x03. Proměnná Bank_A je tedy objektem I^2C slave zařízení, se kterým je komunikace uvažována prostřednictvím I kompatibilního gadgeteer slotu 2.

```
var i2c_gadgeteer_socket = Socket.GetSocket(2, true, null, null);
var Bank_A = new I2CBus(i2c_gadgeteer_socket, 0x03, 100, null);
```

Některé zařízení lze zapojit do různých portů sběrnice BankA a je potřeba definovat adresy s hodnotami jednotlivých portů.

```
byte  BAS1_senzor_LOW = 0x8C, BAS1_senzor_HIGH = 0x8D,
      BAS2_senzor_LOW = 0x8E, BAS2_senzor_HIGH = 0x8F;
```

V tabulce 4.7 jsou uvedeny piny Arduino sběrnice s jejich významem při zapojení prostřednictvím rozhraní této sběrnice. Této metody se v úloze nepodařilo využít a k dispozici jsou tedy všechny piny Arduino sběrnice počítače FEZ Cerbuino Bee. NXShield je osazen čtyřmi tlačítky, kdy tři jsou uživatelsky přístupny prostřednictvím jednotlivých digitálních signálů pro každé z nich. V nezamáčknutém stavu nabývá hodnota logické úrovně signálu stavu true a při zmáčknutí tlačítka naopak false, kdy je navíc prostřednictvím vlastní režie NXShield-Dx rozsvícena příslušná dioda. Prostřednictvím signálů určených pro přenos stavu tlačítek lze tedy také rozsvěcovat a zhasínat barevné diody vývojové desky. V uvažované aplikaci jsou piny NXShield-Dx pro jednotlivá tlačítka dle tabulky 4.7 propojeny s piny jednodeskového počítače FEZ Cerbuino Bee, kdy tlačítko GO je zapojeno do pinu A2, tlačítko LEFT do pinu A3 a tlačítko RIGHT do pinu A1. V tabulce 4.7 jsou pro jednotlivá tlačítka také uvedeny druhy barevných diod, které jsou rozsvíceny při logickém stavu signálu false.

```
OutputPort tl_cervena_LEFT = new
     OutputPort((Cpu.Pin)GHI.Hardware.FEZCerb.Pin.PC3, true);
OutputPort tl_zelena_GO = new
     OutputPort((Cpu.Pin)GHI.Hardware.FEZCerb.Pin.PB0, true);
OutputPort tl_modra_RIGHT = new
     OutputPort((Cpu.Pin)GHI.Hardware.FEZCerb.Pin.PA5, true);
```

Pro přístup ke stavům tlačítek a světelných diod NXShield-Dx jsou deklarovány piny jako objekty třídy OutputPort jmenného prostoru Microsoft.SPOT.Hardware, který je součástí vývojové platformy .NET Micro Framework. Vstupem pro tyto objekty jsou objekty pinů arduino sběrnice FEZ Cerbuino Bee třídy Cpu.Pin ze shodného jmenného prostoru. Jako druhý parametr při tvorbě objektů OutputPort je inicializační logická hodnota signálu. Všechna tlačítka jsou takto deklarována v nezamáčknutém stavu.

Funkce pro nastavení parametrů dle požadovaného řízení motoru

Dle tabulek 4.2 až 4.5 lze vyhodnotit a nastavit mnohé varianty metod řízení servomotoru. V uvažovaných demonstrativních příkladech je využit pouze jeden motor a není tedy potřeba definovat funkce s uvažováním synchronizovaného chodu. Jednotlivé funkce jsou definovány pro uvedení motoru do stavu volnoběhu nastaveného smyslu otáčení při nastavené rychlosti, zastavení chodu motoru a nastavení natočení motoru. Rychlost natočení lze nastavit v rozsahu 0 až ±100 pro maximální chod v kladném nebo záporném smyslu. Dle knihoven vývojářů NXShield-Dx je vhodné tento rozsah nastavit v rozsahu 0 až maximálně ±90 (OpenElectrons, 2014). Pro aplikace je také důležité indikovat aktuální stavy motoru jako například míru natočení motoru nebo rychlost otáčení.

```
public int rychlost_limit(int rychlost)
        {if (rychlost < -90) rychlost = -90;
            if (rychlost > 90) rychlost = 90;
            return rychlost; }
```

Uvedená funkce slouží pro převod libovolné celočíselné hodnoty představující požadovanou rychlost otáčení motoru na hodnotu, která nepřesahuje doporučené extrémní hodnoty. V následujících uvedených funkcích definujících nastavení a uvedení chodu motoru je normalizační funkce vždy volána.

```
public void volnobeh(I2CBus Bank_A,int rychlost,int doba)
        { rychlost = rychlost_limit(rychlost);
         byte[] prikaz = new byte[5];
            prikaz [0] = 0x46;
            prikaz [1] = (byte)rychlost;
            prikaz [2] = (byte)doba;
            prikaz [3] = 0;
            prikaz [4] = 0xC1; //ohled na rychlost, dobu nataceni
            linka.Write(prikaz, 1000); }
```

Výše uvedený kód je určen pro uvedení motoru do chodu s ohledem na požadovanou rychlost a dobu chodu. Nejprve je volána normalizační funkce, pomocí které je požadovaná rychlost převedena na přípustnou hodnotu. Poté je deklarována a naplněna proměnná typu byte[] hodnotami, představující parametry chodu motoru dle tabulek 4.2 a 4.4. Proměnná prikaz tedy představuje zásobník ve smyslu jednorozměrného pole typu byte. Jelikož je každý prvek pole o rozměru byte a stejně tak rozměrné jsou velikosti

jednotlivých adres, je možné obsah zásobníku zapsat do několika adres jednou funkcí, kdy na každou následující adresu je zapsán následující prvek zásobníku. První byte zásobníku představuje adresu, na kterou je zapsán následující obsah pole. Tento princip je deklarován metodou Write(), pocházející z namespace Gadgeteer.Interfaces. Následující prvky zásobníku jsou tedy zapsány do registrů, počínajícího adresou uvedenou v prvním poli zásobníku, kdy dle tabulky 4.2 tato hodnota představuje nastavenou rychlost otáčení. Další adresa je určena pro hodnotu nastavení doby běhu motoru v sekundách, poté následuje adresa registru pro příkaz B a poslední hodnota představuje příkaz A dle principu uvedeného v tabulce 4.4. Hexadecimální hodnota 0xC1 je po převodu do binárního tvaru 11000001, což dle tabulky 4.4 značí nastavení řízení nastavené rychlosti otáčení, otáčení po nastavenou dobu a vlastní uvedení motoru do chodu.

Dále jsou uvedeny dvě funkce využívající nastavení natočení prostřednictvím vestavěného enkodéru. Pokud je režim chodu motoru nastaven s ohledem na dobu chodu, tak má doba chodu pří řízení otáčení větší prioritu a motor se může zastavit dříve, než dosáhne požadovaného natočení. Proto jsou následující funkce navrženy tak, aby byl motor natáčen do určité polohy s ohledem na řízení rychlosti natáčení. Jako první je uvedena funkce pro natočení na absolutní hodnotu enkodéru.

```
public void natocit_na(I2CBus linka, int rychlost, long natoceni)


        {rychlost = rychlost_limit(rychlost);
         byte[] enkoder = new byte[4];
         enkoder = long_to_byte(natoceni);
         byte[] command = new byte[9];
         command[0] = 0x42;
         command[1] = enkoder[0];    //stupne
         command[2] = enkoder[1];
         command[3] = enkoder[2];
         command[4] = enkoder[3];
         command[5] = (byte)rychlost;
         command[6] = 0;
         command[7] = 0;
         command[8] = 0x89;
         linka.Write(command, 1000);}
```

Funkce natocit_na() obdobná s funkcí volnobeh(), kdy jediným rozdílem jsou rozdílné vstupní parametry, funkce převádějící požadovanou hodnotu natočení typu long o velikosti čtyř byte na čtyři prvky pole o rozměru byte, tvar příkazu pro chod motoru a výsledný rozměr zásobníku. Tvar příkazu v hexadecimálním tvaru 0x89 lze interpretovat v binárním tvaru jako řetězec bitů 10001001, což dle tabulky 4.4 znamená nastavení chodu motoru s ohledem na řízení rychlosti natáčení, dosažení absolutního natočení dle nastavené hodnoty enkodéru a následné uvedení motoru do chodu. Jako parametry byla odebrána předávaná požadovaná hodnota nastavení rychlosti a nahradila ji požadovaná hodnota absolutního natočení motoru datového typu long o velikosti čtyř byte. Do jednotlivých

registrů pro hodnotu natočení enkodéru lze zapisovat po jednom byte a typ long je tedy nutné převést na pole se čtyřmi prvky typu byte. Za tímto účelem byla vytvořena funkce long_to_byte, která za pomocí celočíselného dělení přijme hodnotu typu long a navrátí čtyřrozměrné pole hodnot typu byte.

```
public byte[]  long_to_byte(long cislo)
        {byte[] long_byte = new byte[4];
         long_byte[0] = (byte)(int)(cislo / 1);
         long_byte[1] = (byte)(int)(cislo / 255);
         long_byte[2] = (byte)(int)(cislo / 255 / 255);
         long_byte[3] = (byte)(int)(cislo / 255 / 255 / 255);
         return long_byte; }
```

V poli s indexem 0 je uložena hodnota LSB, pro kterou je uvažován registr s adresou 0x42 pro slot motoru 1 a obdobně v poli s indexem 3 je uložena hodnota MSB požadovaného absolutního natočení enkodéru.

Funkce pro relativní natočení motoru je shodná s funkcí pro absolutní natočení s jediným rozdílem, který spočívá ve tvaru příkazu zapisovaného na adresu 0x49, jehož hodnota v binární podobě musí mít dle tabulky 4.4 nastaven třetí bit zprava na 1. Příkaz má pak v hexadecimálním tvaru hodnotu 0x8D a jako řetězec bitů 10001101.

Dále je potřeba definovat funkci, prostřednictvím které bude možné motor nechat zabrzdit. Zabrzdění motoru lze dosáhnout zapsáním příkazu o velikosti byte dle tabulky 4.5 na adresu 0x41. Pro zastavení motoru je vybrán příkaz s hodnotou v hexadecimálním tvaru 0x41, kdy je zastavení chodu se zapnutou brzdou.

```
public void zabrzdi(I2CBus linka)
        {byte[] command = new byte[5];
         command[0] = 0x41;
         command[1] = 0x41;
         linka.Write(command, 1000); }
```

Další uvedená funkce slouží pro navrácení hodnoty enkodéru datového typu long. Funkci lze volat pro motor zapojený do jednoho ze dvou portů určených pro zapojení servomotoru. Nejprve jsou dle zadaného parametru zapojení vyhodnoceny adresy s jednotlivými byte, ze kterých je následně po jejich načtení z příslušných registrů vypočtena a navrácena hodnota natočení enkodéru datového typu long.

```
public long  natoceno_na(I2CBus linka, string motor_port)
        {
           byte[] adresa = new byte[4];
           if (motor_port == "Motor-1") { adresa[0] = 0x62;
                adresa[1] = 0x63; adresa[2] = 0x64; adresa[3] =
                0x65; }
```

```
        else { adresa[0] = 0x66; adresa[1] = 0x67; adresa[2] =
        0x68; adresa[3] = 0x69; }
        byte[] natoceni0 = new byte[1]; byte[] natoceni1 = new
        byte[1]; byte[] natoceni2 = new byte[1]; byte[]
        natoceni3 = new byte[1];
    linka.WriteRead(new byte[] { adresa[0] }, natoceni0, 1000);
    linka.WriteRead(new byte[] { adresa[1] }, natoceni1, 1000);
    linka.WriteRead(new byte[] { adresa[2] }, natoceni2, 1000);
    linka.WriteRead(new byte[] { adresa[3] }, natoceni3, 1000);
    long natoceni = ((int)natoceni0[0] + ((int)natoceni1[0] *
        255) + ((int)natoceni2[0] * 255 * 255) +
        ((int)natoceni3[0] * 255 + 255 * 255));
    return natoceni; }
```

Senzor dotyku

Senzor dotyku je schopen indikace dvou vlastních stavů. Jeho naměřená hodnota ukládaná do registru je po převedení na celočíselnou hodnotu 253 v nezamáčknutém stavu a přibližně 179 ve stavu zamáčknutém. Je třeba brát ohled na skutečnost, že pokud je porucha na komunikační lince a není možné přistupovat k hodnotám registrů NXShield-Dx, tak se přijímaná hodnota rovná indikační hodnotě proměnných, nule. Stav senzoru může být tedy indikován jako zamáčknut, nezamáčknut a chyba komunikace. Je tedy vhodné vytvořit funkci, která navrátí celočíselnou hodnotu 0 pro chybu komunikace, 1 pro stav nezamáčknutý a 2 pro stav zamáčknutý.

```
public int Dotyk(I2CBus Bank_A,string port)
        {byte[] hodnota = new byte[1];
         int adresa = 0x8C;
         if (port == "BAS1") {adresa = BAS1_senzor_LOW ;} else
         adresa = BAS2_senzor_LOW;
         Bank_A.WriteRead(new byte[] { (byte)adresa }, hodnota,
           1000);
         Debug.Print("SENSOR1LSB: " + hodnota[0].ToString());
         int zmacknuto = 0;
         if (hodnota[0] > 100 && hodnota[0] < 200)
            { zmacknuto = 2; } else { zmacknuto = 1; }
            if (hodnota[0] == 0)    { zmacknuto = 0; }
            return zmacknuto ;}
```

Volání funkce:

```
int je_zamacknuto = Dotyk(linka, "BAS1");
```

Světelný senzor (režim snímání intenzity světla)

Při zapojení světelného senzoru do portu BAS2 snímá v režimu měření intenzity dopadajícího světla. Při tomto zapojení je také potřeba přistupovat k hodnotám obou registrů naměřených hodnot senzoru portu BAS2_HIGH a BAS2_LOW dle předem nadefinovaných adres.

```
public int intenzita(I2CBus linka)
      {int intenzita = 0;
      byte[] hodnotaLSB = new byte[1];
      byte[] hodnotaMSB = new byte[1];
      linka.WriteRead(new byte[] { BAS2_sensor_LOW }, hodnotaLSB,
      1000);
      linka.WriteRead(new byte[] { BAS2_sensor_HIGH }, hodnotaMSB,
      1000);
      Debug.Print("BarvaLSB: " + hodnotaLSB[0].ToString() + "
      Barva1MSB: " + hodnotaMSB[0].ToString());
      int barva_int = ((int)hodnotaLSB[0] + ((int)hodnotaMSB[0] *
      255));     return intenzita; }
```

Tímto byly definovány funkce pro komunikaci se všemi disponovanými moduly stavebnice LEGO Mindstorms Education. Následující demonstrativní úlohy jsou příkladem užití uvedených funkcí

5.3 Demonstrativní úlohy

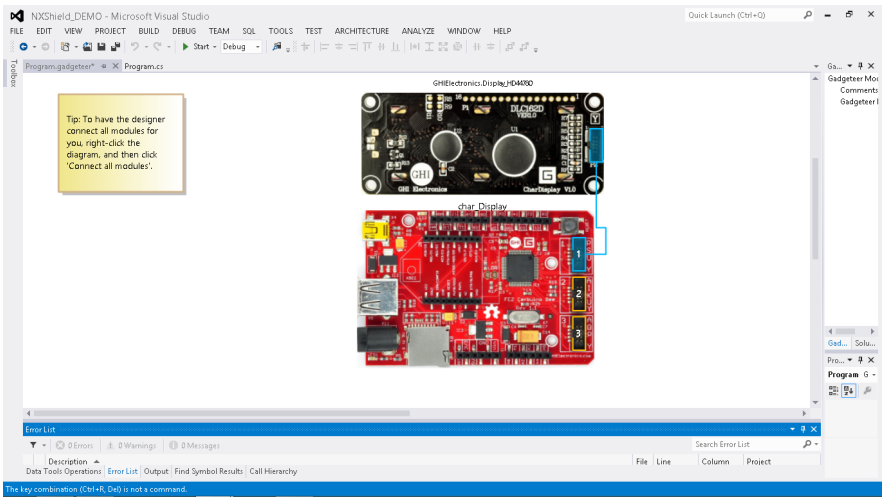
Pro tvorbu demonstrativních úloh jsou jako periferie k dispozici moduly stavebnice LEGO MIndstorms Education velký servomotor, senzor dotyku, barevný senzor a moduly GHI Gadgeteer char displej HD44780 a senzor teploty a vlhkosti. Všechny zmíněné moduly lze mít zapojeny a využívat zároveň, kdy moduly GHI jsou zapojeny do gadgeteer slotů 1 a 3 a moduly LEGO MIndstorms Education jsou zapojeny dle zvolené demonstrativní úlohy.

Výsledný software je uvažován jako soubor všech vytvořených úloh, kdy prostřednictvím úlohy 1, uživatelského grafického rozhraní, jsou uváděny do chodu další programy. Nejprve je tedy představena úloha grafického rozhraní, kdy jsou uživateli prostřednictvím GHI char displeje zobrazovány informace a prostřednictvím tlačítek NXShield-Dx lze zprostředkovávat vstupy uživatelem. Pro možnost využití GHI Gadgeteer modulů jsou úlohy navrženy ve vývojové platformě .NET Gadgeteer, kdy jsou využívány metody jmenných prostorů vývojové platformy .NET Gadgeteer a .NET Micro Framework.

Uživatelské rozhraní navrženého software

Uživatelské rozhraní je základem pro předávání informací o požadovaných vstupech a probíhajících procesech uživateli. Funkce uživatelského rozhraní by měly být aktivní a schopny vyhodnocovat uživatelské vstupy po celou dobu chodu programu.

Jednotlivé nadefinované programy jsou prováděny v rámci cyklů. Při užití vývojové platformy .NET Gadgeteer jsou cykly zprostředkovávány pomocí časovačů. Při inicializaci není vhodné použít nedoporučené cyklické funkce, jako například while(true), protože platforma Gadgeteer neumožňuje v tomto případě přerušení pro režii GHI Gadgeteer modulů a časovačů. Nejprve je tedy založen nový Gadgeteer projekt a definován časovač s periodou spuštění 500 ms, prostřednictvím kterého bude aktualizován displej a přijímány vstupy tlačítky vývojové desky NXShield-Dx. Nově založený projekt obsahuje dvě části, základní třídu Program, do které jsou vloženy deklarace a metoda této třídy ProgramStarted, která je určena pro definici časovačů a GHI modulů. Aby bylo možné užívat GHI displej, je nutné nastavit v grafickém prostředí projektu jeho zapojení do Gadgeteer slotu jednodeskového počítače. Zapojení displeje je nastaveno do Gadgeteer slotu 1 a jeho nastavení v grafickém rozhraní vývojové platformy je zobrazeno na obrázku 5.1.



Obr. 5.1: Grafické rozhraní .NET Gadgeteer s nastavením zapojení GHI Char Displeje HD44780

Hlavní vlákno programového kódu se nachází v souboru Program.cs. Navržený zdrojový kód, přiložený jako příloha A, představuje fázi software, kdy jsou deklarovány vyžadované nezákladní jmenné prostory, proměnné pro přístup ke stavům tlačítek a časovač s periodou 500 ms, který v této fázi nemá naprogramovanou žádnou funkci. Zmíněný časovač je objektem s názvem GUI třídy Gadgeteer.Timer. V části programu, která je určena pro inicializační funkce, je také naprogramované zapnutí podsvícení použitého displeje a smazání jeho zobrazovaného obsahu. Deklarace referencí jmenných

prostor pro možnost užívání GHI modulů jsou automaticky generovány po nastavení zapojení vlastních modulů dle obr. 5.1.

Program s hotovým uživatelským rozhraním je uveden v rámci přílohy B a jeho jednotlivé ukázky jsou na obrázku 5.2. V rámci uvedeného programu je nejprve po uživateli prostřednictvím GHI displeje vyžadován stisk tlačítka GO na vývojové desce NXShield-Dx. Po stisknutí tlačítka se program přesouvá do dalšího kroku, kdy je na druhém řádku vypisován název zvolené úlohy. Volbu úlohy lze modifikovat prostřednictvím stisknutí tlačítek LEFT nebo RIGHT na vývojové desce. Potvrzení volby úlohy pro její spuštění je provedeno pomocí stisku tlačítka GO. Za provedení stisku tlačítka je prostřednictvím navrženého algoritmu považováno pouze počáteční zmáčknutí. Pokud je tlačítko drženo v zamáčknutém stavu a je očekáván jeho stisk, tak stisk není kladně vyhodnocen dokud není tlačítko zamáčknuto z nezamáčknutého stavu. Při držení tlačítka GO po dobu pěti vteřin je program uveden do původního stavu, kdy očekává stisk tlačítka GO před volbou úlohy pro spuštění. Pro přidání volby nové úlohy do uživatelského rozhraní je nutné pouze přidat textový řetězec s názvem úlohy v rámci proměnné uloha_nazev typu pole stringů a také nastavit číselnou hodnotu počtu úloh prostřednictvím celočíselné proměnné pocet_uloh.

Obr. 5.2: Ukázky grafického rozhraní navrženého software v rámci fotografií GHI Char Displeje HD44780

Novými prvky v inicializační části programu jsou globální proměnné typů integer, bool a pole stringů, pomocí kterých jsou v rámci celého programu předávány informace o stavu programu, tlačítek a volbě a počtu naprogramovaných úloh. Proměnné s předponou GUI slouží pro uchování informací o uživatelském rozhraní. Hodnota proměnné s příponou krok definuje, v jakém z navržených kroku se momentálně nachází program. Program, uvedený v příloze B, při hodnotě proměnné s příponou krok rovné 0 čeká na stisk tlačítka, při hodnotě rovné 1 je požadována volba jedné z úloh pro spuštění a potvrzení stiskem

tlačítka GO, kdy po vlastním potvrzení program přechází do stavu chodu zvolené úlohy a hodnota proměnné krok se rovná 2.

Proměnná s příponou volba slouží pro uchování hodnoty představující pořadí zvolené úlohy v rozsahu od nuly do celkového počtu úloh menšího o jedna. V rámci modifikace volby úlohy jsou hlídány hodnoty proměnné s ohledem na počet úloh nastavený prostřednictvím hodnoty proměnné pocet_uloh. Proměnná s příponou reset slouží pro uchování informace o počtu vteřin, po který je drženo tlačítko GO. Časovač, prostřednictvím kterého je tato hodnota inkrementována, je v uvedeném případě deklarován s periodou spuštění půl vteřiny, za vteřinu se tedy při zamáčknutém tlačítku GO zvýší hodnota proměnné GUI_reset o 2.

V hlavním vláknu programu, funkci ProgramStarted, jsou v rámci GUI vytvořeny dva objekty časovačů. Časovač s názvem tlacitka je užíván pro indikaci, kdy libovolné tlačítko změní svůj stav z nezamáčknutého na zamáčknutý. Uchovávány jsou stavy tlačítek při minulém spuštění časovače a pokud je logická hodnota představující zamáčknutí dle zmíněného principu invertována, je indikováno momentální zmáčknutí tlačítka a příslušná proměnná pro tlačítko s příponou stisk je nastavena na hodnotu true. Čtení stavů tlačítek a zhodnocení jejich zmáčknutí je v uvedeném programu nastaveno s periodou provedení desetiny sekundy.

Časovač s názvem GUI je v rámci objektu vytvořen s periodou spínání co půl sekundy a při jeho spuštění je na základě hodnoty globální proměnné GUI_krok překreslen obsah displeje dle navrženého algoritmu.

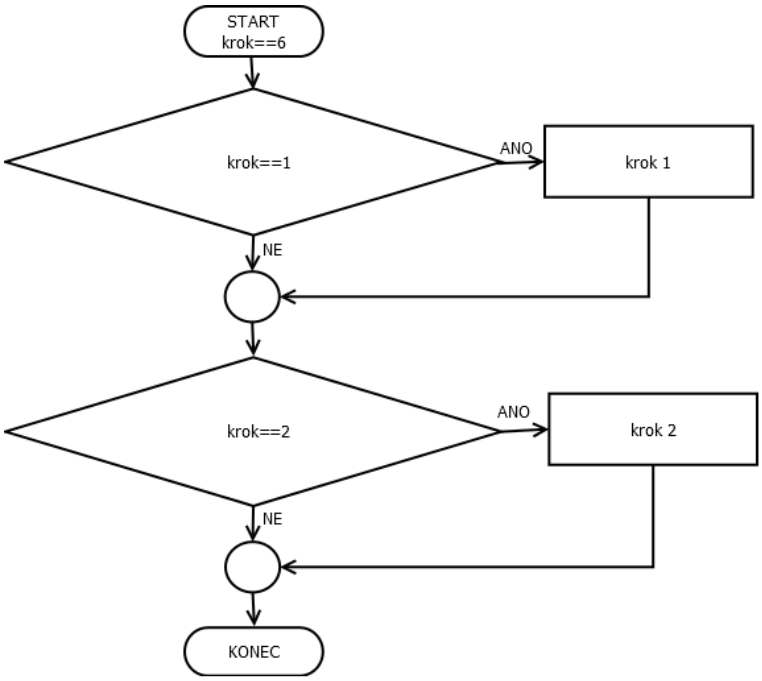
Pokud se při volání metody spuštění časovače program nachází v kroku nula, je smazán obsah displeje, na první řádek od prvního sloupce je vypsán text "Zmackni tlacitko" a pak je od osmého sloupce na druhém řádku vypsán text "GO". Pokud bylo před průchodem podmínkou stisknuto tlačítko z nezamáčknutého stavu, hodnota proměnné GUI_krok je inkrementována a logická proměnná představující zamáčknutí tlačítka je nastavena na hodnotu false, aby při průchodu následující podmínkou nebylo okamžitě provedeno potvrzení v následujícím kroku. Proto je v každém kroku nutné opětovně zamáčknout tlačítko.

Pokud se při průchodu metody spuštění časovače objektu GUI nachází program v kroku 1, tak je smazán displej, na první řádek je vypsán text " Zvol program->GO" a na druhý řádek displeje je vypsán název zvolené úlohy dle hodnoty string obsažené v poli proměnné uloha_nazev pod indexem s hodnotou rovnající se volbě úlohy. Je také naprogramována režie hlídání přetečení hodnoty volby úlohy dle nastavené hodnoty proměnné pocet_uloh. Pokud byl indikován stisk tlačítka GO, hodnota proměnné krok je inkrementována a hodnotu proměnné indikující stisknutí tlačítka GO je z již uvedeného důvodu nastaven na false. V dalším kroku je již prováděna zvolená úloha.

Jak již bylo zmíněno, grafické rozhraní i s chodem programu lze uvést do původního stavu prostřednictvím držení tlačítka GO v zamáčknutém stavu po dobu pěti sekund. Tato funkce je zprostředkována v rámci průchodu poslední podmínkou časovače

GUI. Tato podmínka musí být v časovači poslední, nezáleží u ní tudíž na kroku, ve kterém se program nachází (resetovat lze tedy v libovolné fázi chodu programu) a obsahuje funkci dopsání číselného odpočtu do resetování na displej. Kdyby nebylo zmíněné dopsání provedeno na konci průchodu časovačem, bylo by přepsáno při mazání displeje při průchodu podmínkami týkajících se aktuálního kroku programu.

Program, uvedený jako příloha B, je ve stavu, kdy lze zvolit úlohy na základě názvů, avšak žádné úlohy nejsou v této verzi implementovány. Hotový program se všemi navrženými úlohami je uveden v rámci přílohy obsažené na přiloženém CD, které je součástí vazby diplomové práce. Hotový software obsahuje navíc funkce pro režii zapojených zařízení LEGO Mindstorms Education definované na začátku kapitoly a funkce, ve kterých jsou naprogramovány navržené úlohy. Před spuštěním každé jednotlivé úlohy je v jednotlivých funkcích úloh naprogramována inicializační fáze, kdy jsou uživateli prostřednictvím displeje zobrazovány informace ohledně volitelného nastavení zapojení periferií a po potvrzení obrazovky s rekapitulací zvolených parametrů zapojení program běží v rámci zvolené úlohy. Chod úlohy je možné ukončit podržením tlačítka GO po dobu pěti sekund. Po odsouhlasení spuštění zvolené úlohy stiskem tlačítka GO je inkrementován krok GUI na hodnotu 2, což má za následek, že časovač stále běží, ale prostřednictvím něj lze pouze prostředí resetovat. Inkrementována je také proměnná uloha_krok, prostřednictvím které je potom vyhodnocováno, která fáze úlohy je momentálně prováděna. Algoritmus principu právě spouštěného kroku úlohy je uveden na v rámci schématu uvedeného na obrázku 5.3.



Obr. 5.3: Schéma algoritmu, který vyhodnocuje aktuální krok úlohy při spuštění Časovače úlohy

Úloha využívající servomotor a senzor dotyku

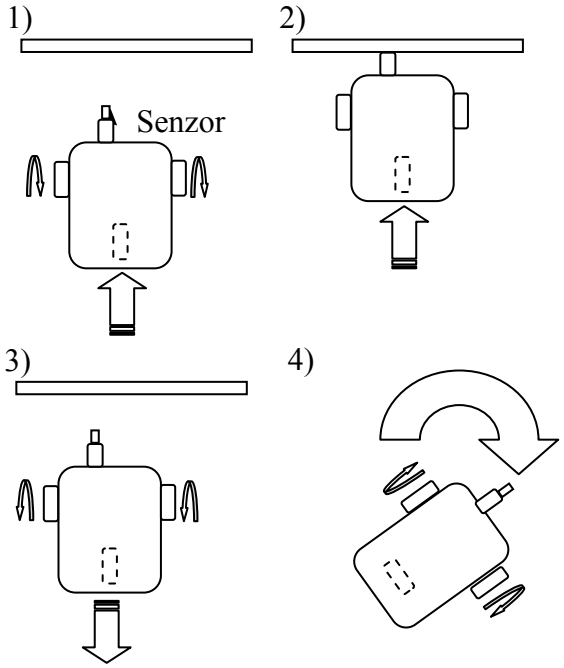
Jedná se o úlohu, kdy je uvažovaná soustava uváděna do pohybu jedním nebo dvěma motory a je zkoumána detekce kolize prostřednictvím jednoho či dvou senzorů dotyku. V případě, že je vyhodnocena uvažovaná kolize zamáčknutím nejméně jednoho senzoru dotyku, úloha přejde do následující fáze vzhledem k počtu navolených zapojených motorů. Pokud je zapojen pouze jeden motor a soustava je schopna vykonávat pohyb pouze dopředu nebo dozadu bez možnosti řízeného otočení, tak je motor zastaven a pro opětovné spuštění je nutné ho manuálně odstranit z kolizní dráhy. Při užití dvou motorů je naprogramována funkce mírného odjezdu a následného otočení od překážky, po kterém je soustava opět uvedena do smyslu přímočarého pohybu. Schéma představující jednotlivé fáze úlohy po volbě zapojení je uvedeno na obrázku 5.3.

Po zvolení libovolné úlohy ke spuštění je v rámci volání objektu časovače úloh cyklicky spouštěna funkce s naprogramovanou úlohou dle principu podmínky switch s ohledem na hodnotu volby specifické úlohy. Ve funkci každé úlohy je obdobný přístup, jak tomu je u jednotlivých kroků GUI. Tento algoritmus je zobrazen v rámci schématu na obrázku 5.3.

V prvním a druhém kroku této úlohy jsou uživateli vypisovány informace o požadovaném zapojení senzoru dotyku do jednoho z podporovaných portů sběrnice Bank A. Algoritmus volby portu pro zapojení je uveden v příloze C. Jelikož je senzor dotyku podporován portem BAS1 i BAS2, pro možnost volby zapojení byl naprogramován kód, pomocí kterého lze port zvolit a jeho název je vypisován na druhý řádek displeje. Volba je provedena stiskem tlačítka GO, kdy v následujícím druhém kroku je obsažen kód plnící obdobnou funkci, avšak s ohledem na zapojení motoru. Po potvrzení zapojení motoru do jednoho z portů Motor-1, Motor-2 nebo dvou motorů do obou portů je program úlohy posunut do kroku třetího, kdy jsou uživateli vypsány zvolené parametry zapojení a pro započetí vlastního chodu úlohy je vyžadováno stisknutí tlačítka GO.

V posledním kroku úlohy jsou nadeklarovány objekty tříd I^2C komunikace a prostřednictvím dříve nadefinované funkce Dotyk() je navrácena celočíselná hodnota představující výsledek komunikace s účelem načtení dat naměřených senzorem dotyku. Při užití dvou senzorů je nutné vyhodnotit jejich vzájemnou interakci, kdy pokud alespoň jeden ze senzorů neposkytuje naměřené informace, je vyhodnocena chyba komunikace, jinak pokud jsou oba zároveň nezamáčknuty, je vyhodnocen stav bez kolize a pokud ani tato podmínka neplatí tak je vyhodnocen stav kolize. Nyní je prostřednictvím podmínky switch definována patřičná akce dle načtené hodnoty. Pokud je hodnota vyhodnocená z informací přijatých ze senzoru rovna nule, znamená to, že komunikace neproběhla nebo senzor není zapojen ve zvoleném portu. V tomto případě je vypsáno hlášení o chybě čtení dat a bez ohledu na to, jestli byl motor v chodu je do vývojové desky odeslán příkaz pro jeho zastavení. Pokud je funkcí Dotyk() navrácena hodnota rovna jedné, senzor je vyhodnocen jako nezamáčknutý, je vypsáno hlášení a motor je uveden do provozu při poloviční maximální rychlosti v dopředném smyslu. Předpokladem je, že objekt poháněn motorem se pohybuje ve směru, kde když dojde ke kolizi, tak bude zamáčknut senzor

dotyku. V tomto případě je funkcí Dotyk() navrácena hodnota rovna dvěma, při jejíž přijetí je vypsáno hlášení o kolizi a v případě volby zapojení pouze jednoho motoru je motor zabrzděn prostřednictvím definované funkce Zabrzdi. Algoritmus vyhodnocení kolize je uveden v příloze D.



Obr. 5.4: Schéma úlohy využívající senzor dotyku znázorňující pohyby soustavy při kolizi

Za předpokladu volby zapojení obou motorů je v rámci úlohy naprogramováno více funkcí, protože lze prostřednictvím nastavení různé rychlosti otáčení jednotlivých motorů dosáhnout natáčení poháněné soustavy. Pokud je tedy nastaveno, že mají být zapojeny oba motory, soustava je po odsouhlasení rozpohybována ve smyslu dopředného chodu. Při každém spuštění časovače s úlohou, je načítán aktuální stav senzoru dotyku a pokud je jeho stav vyhodnocen jako zamáčknutý, úloha je převedena do fáze otočení. Předpokladem je, že soustava se při stavu kolize nachází přímo u dosažené překážky a z tohoto důvodu je v první fázi otočení zahájen zpětný chod pětinovou maximální rychlostí po dobu deseti cyklů časovače. Vzdálenost, o kterou se za tuto dobu vzdálí soustava od překážky, je závislá na době trvání jednoho cyklu úlohy vzhledem k nastavené hodnotě spouštění časovače, obvodu hnacích kol a případných zaseknutí pohyblivých částí soustavy bránících v pohybu. Při provozu úlohy na fyzické soustavě je tedy třeba tyto fáze pohybu ladit, aby byla soustava oddálena od překážky do té míry, kdy bude mít dostatečný prostor na vlastní otočení. Algoritmus odjezdu od kolize je uveden v příloze E.

Po předpokládaném vzdálení od dosaženého místa kolize jsou dvě možnosti jak dosáhnout otočení soustavy. Pokud by byl jeden z motorů zabrzděn a druhý uveden do provozu, docházelo by k rotaci kolem osy kola zastaveného motoru. Jako vhodnější byla zvolena metoda otočení soustavy, kdy jsou uvedeny oba motory do stavu chodu navzájem opačného smyslu při pětinové maximální rychlosti. U této metody se osa rotace nachází mezi hnacími koly, která je blíže středu soustavy a vozidlo se takto při otáčení méně

vzdaluje a je menší šance kolize při otáčení. Tato fáze probíhá také po nastavený počet cyklů úlohy a je potřeba nastavit dle fyzické soustavy počet cyklů tak, aby došlo k otočení soustavy o přibližně 180°. Po předpokládaném otočení jsou vynulovány proměnné definující počet spuštěných cyklů a úloha je převedena do stavu, kdy se pohybuje dopředu než narazí na další překážku. V rámci odjezdu od překážky není brán ohled na informace naměřené senzorem a jeho stav je vyhodnocován až po otočení od překážky a zahájení opětovného pohybu v kladném smyslu.

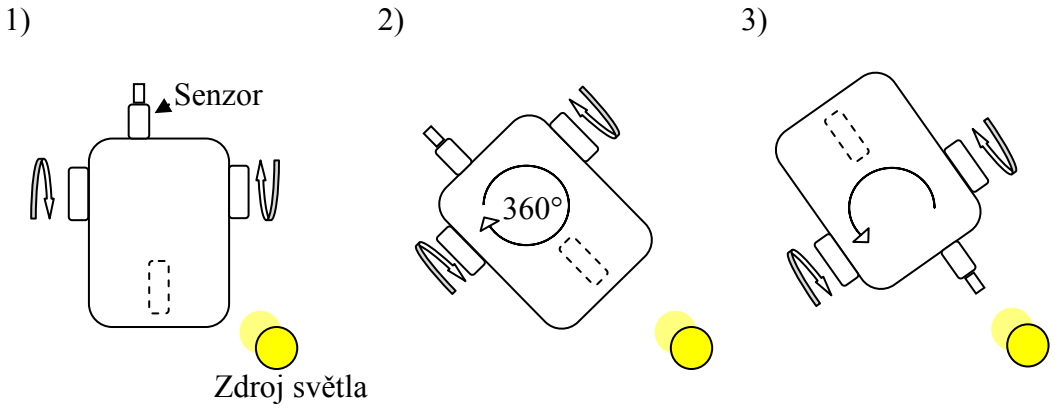
Při úlohách, kde je soustava do pohybu uváděna v rámci programového řízení a jsou uvažovány kolize, je nutné brát ohled na maximální nastavenou rychlost pohybu, vůči které by v případě okamžitého zastavení působilo na soustavu úměrné zrychlení a tento ráz by mohl zapříčinit následnou destrukci některých z komponent. Proto jsou v rámci těchto úloh voleny rychlosti v každém kroku maximálně 22% nastavitelného maxima.

Chod úlohy lze ukončit naprogramovaným restartováním při podržením tlačítka GO po dobu pěti sekund. Funkce volaná pro reset programu byla kvůli úloze rozšířena o funkci Zabrzdi() pro případ, kdyby byl během resetu motor v chodu a také proměnné definující volby zapojení a čítače cyklů jsou uvedeny na původní hodnoty.

Úloha využívající servomotor a světelný senzor v režimu snímání světelné intenzity

V druhé úloze není uvažován pohyb soustavy z místa, pouze její natáčení do směru, ze kterého je naměřena největší světelná intenzita. Princip úlohy je převzat z realizace otočných solárních panelů, které se automaticky natáčí aby bylo prostřednictvím dopadajícího světla pod ideálním úhlem generováno maximální možné napětí.

Po spuštění druhé úlohy nastává inicializační fáze, týkající se volby zapojení periferií, která je obdobná s úlohou předchozí. Senzor světla lze však využít v režimu snímání světelné intenzity pouze při zapojení do portu BAS2 a první krok úlohy je čistě krokem informativním, kdy je uživateli vypsána informace o požadovaném zapojení světelného senzoru. V této úloze není uvažován jiný pohyb, než natáčení soustavy, proto je volba zapojení motorů volena pouze jako Motor-1 nebo Motor-2.



Obr. 5.5: Schéma úlohy využívající senzor světla v režimu měření světelné intenzity

Ve čtvrtém kroku, který představuje chod samotné úlohy, je vypsáno hlášení o chování soustavy, jejím natočení a indikované hodnotě naměřené světelné intenzity. Tyto zobrazované informace musí být před samotným výpisem přijaty prostřednictvím I^2C komunikace a pokud není soustava natočena do pozice, kde byla naměřena největší hodnota světelné intenzity je zprostředkováno natočení do této pozice maximální rychlostí. Hodnota natočení enkodéru a maximální naměřené hodnoty světelné intenzity je při inicializaci nastavena na hodnotu 0, pro započetí nového měření je nutné stisknout tlačítko GO, kdy program úlohy přechází do dalšího kroku. Měření intenzity probíhá v rozsahu 360° s krokem přibližně 10°, kdy je vždy načteno aktuální natočení enkodéru, je porovnáno, jestli od počátečního natočení neproběhlo jedno otočení kolem osy a pokud ne, tak je provedeno natočení o úhel jednoho kroku. Po vlastním natočení je načtena naměřená světelná intenzita a pokud je její hodnota větší, než byla největší naměřená hodnota tak je její hodnota uložena do proměnné s maximální naměřenou intenzitou. S maximální hodnotou intenzity je také potřeba ukládat hodnotu natočení enkodéru. Uživateli jsou vypisovány informace o přijatých hodnotách a chování soustavy. Pokud je indikováno otočení soustavy o více, než 360°, tak je dekrementována hodnota kroku úlohy a soustava je natočena do pozice, kde byla naměřena největší hodnota světelné intenzity. Algoritmus natáčení soustavy na směr, z něhož je naměřena největší dopadající světelná intenzita je uveden v příloze F.

Jednotlivé nastavení, definující otočení v jednom kroku o úhel přibližně 10° a na tomto závislé celkové otočení na měřící cyklus o 360°, je nutné nastavit dle fyzických parametrů soustavy. Natočení je řízeno prostřednictvím hodnoty enkodéru jednoho motoru a po měřícím cyklu je enkodér zpětně natočen na hodnotu, při jejím natočení byla naměřena největší světelná intenzita dopadající na Čidlo světelného senzoru. Je tedy také vhodné brát ohled na rychlost natáčení, aby nedošlo k prokluzu kol a natočení enkodéru bylo přímo úměrné natočení celé soustavy v prostoru. V rámci úlohy není také počítáno s překážkou v dráze jejího rotačního pohybu.

Úloha ovládání IR vysílačem

V této úloze je mimo GHI char displeje užitý také další GHI Gadgeteer modul, IR přijímač. Přijímač je schopen vyhodnocovat signály předávané prostřednictvím infračerveného světla a pomocí vysílače, který byl součástí balení, lze takto posílat 7 různých ovládacích povelů. V rámci úlohy lze tedy uvést soustavu do pohybu ve smyslu dopředu, dozadu, doleva, doprava a pohyb zastavit. Tlačítky vysílače lze také modifikovat rychlost libovolného pohybu.

Obr. 5.6: IR GHI přijímač a vysílač

V první fázi úlohy jsou uživateli na displej vypsány základní informace ohledně ovládání vozidla. Pro chod úlohy jsou vyžadovány zapojené oba motory to portu Motor-1 a Motor-2. Každé z tlačítek vysílače uvedeného na obrázku mají v úloze ovládání vozidla naprogramovanou vlastní funkci. Přijatá informace o stisku tlačítka je v prostředí .NET Gadgeteer vyhodnocována až ve chvíli, kdy je změna jeho stavu ze zamáčknutého změněn na nezamáčknutý. Jeho vlastní držení tedy nemá žádný význam a pro opakování volby je potřeba tlačítko opětovně stisknout a pustit. Červené tlačítko má v úloze naprogramovanou funkci zastavení obou motorů, tlačítka s šipkami o řádek níže uvedou vozidlo do přímočarého pohybu ve smyslu dle stisknuté šipky, přeškrtnutý reproduktor zvyšuje nastavenou rychlost pohybu a poslední tlačítko ve stejném sloupci slouží pro snížení rychlosti pohybu. Šipky směřující doleva a doprava uvádí soustavu do pohybu, kdy bylo potřeba určit, zda se v tuto chvíli má vozidlo pouze natáčet nebo se u toho také pohybovat dopředu, čili zatáčet. Vhodnější je užití funkce zatáčení, kdy má vozidlo podobnou rychlost, jako při přímočarém pohybu a jeho chod poté působí plynuleji, než když u natáčení musí eliminovat svou rychlost, natočit se, a pak může pokračovat v jízdě. Při používání funkce pro změnu rychlosti pohybu je potřeba ještě opakovat volbu pro libovolný pohyb. Změna rychlosti má vliv na přímočarý pohyb a také zatáčení.

Pro možnost použití IR přijímače v projektu .NET Gadgeteer je nutné ho v grafickém rozhraní vývojového prostředí najít v seznamu dostupných modulů a po vložení do projektu ho zapojit do jednoho z podporovaných Gadgeteer slotů osazených na jednodeskovém počítači FEZ Cerbuino Bee. Tento modul je podporovaný všemi sloty typu X a lze jej tedy zapojit do libovolného Gadgeteer konektoru uvažovaného počítače. Po zapojení modulu v grafickém rozhraní vývojového prostředí již lze vytvořit .NET Gadgeteer objekt události přijetí infračerveného signálu. Při zmáčknutí libovolného tlačítka IR vysílače je vyhodnocena celočíselná hodnota představující specifické tlačítko. Nejprve byly tedy zjištěny všechny specifické hodnoty, představující jednotlivá tlačítka a poté jim byly naprogramovány jednotlivé funkce. V rámci události GHI přijímače jsou vyhodnoceny tři různé funkce. Je nutné brát ohled na skutečnost, že v rámci události není vložena žádná podmínka kontrolující v jakém kroku se nachází program a k události vyhodnocení přijetí IR signálu může dojít v libovolné časti chodu programu. Tohoto je s výhodou využito pro možnost zastavení chodu obou motorů na dálku, což se hodí když je vozidlo v pohybu a je potřeba jej zastavit. Druhá funkce zprostředkovaná v rámci globální

události je modifikace rychlosti, která je nastavovaná jako globální proměnná. Funkce snižující rychlost má naprogramovanou podmínku, aby rychlost nebylo možné nastavit na hodnotu menší, než 10% maximální rychlosti. Nastavení této nejmenší přípustné rychlosti má navíc za následek, že při zatáčení se vozidlo otáčí na místě kolem osy mezi jeho hnacími koly. Příčina tohoto efektu je vysvětlena v rámci funkce úlohy níže. Poslední funkcí globální události je uložení celočíselné hodnoty představující stisknuté tlačítko do globální proměnné, dle které jsou dále nastavovány řídicí parametry v rámci cyklicky spouštěné funkce úlohy.

Pokud byla tedy v GUI zvolena třetí úloha s názvem IR_ovl, tak program podmíněně volá funkci třetí úlohy, ve které je na základě hodnoty globální proměnné představující stisknuté tlačítko IR vysílače odeslány řídicí parametry do zařízení NXShield-Dx. Ihned po zvolení úlohy lze již vozidlo ovládat a na displeji si je možné přečíst informace o významu jednotlivých tlačítek ovladače. Na druhém řádku je vypisována hodnota nastavené rychlosti v procentech maximální nastavitelné rychlosti pohybu. Pokud bylo tedy zmáčknuto tlačítko, při volání funkce úlohy je dle hodnoty provedena naprogramovaná akce a hodnota globální proměnné je vynulována aby akce neprobíhala opakovaně, ale pouze když bylo tlačítko stisknuto. Pokud je vyhodnocen ovládací požadavek na přímočarý směr ve smyslu dopředu nebo dozadu, tak jsou oba servomotory asynchronně zapnuty v požadovaném smyslu otáčení. Pokud je stisknuto tlačítko doleva nebo doprava, tak je jeden z motorů uveden do chodu s rychlostí menší o 20% maximální rychlosti otáčení. Vzhledem k tomu, že lze nastavit rychlost pohybu na extrémní hodnotu 10% maximální rychlosti, tak se při tomto nastavení a zároveň požadavku na zatáčení točí vozidlo kolem osy mezi jeho hnacími koly, protože jeden servomotor má nastavenou rychlost 10% a druhy o 20% menší, čili -10%. Při nastavení rychlosti větší, než 20%, jsou oba pohony uvedeny do provozu při stejném smyslu otáčení a vozidlo se při natáčení také pohybuje dopředu.

Úloha hledání zdroje hluku

Tato úloha je zaměřena na hledání a dosažení polohy zdroje hluku s vetší nebo stejnou intenzitou, než je naměřený referenční vzorek. V úloze jsou využity dva velké LEGO MIndstorms Education servomotory a jeden senzor zvuku.

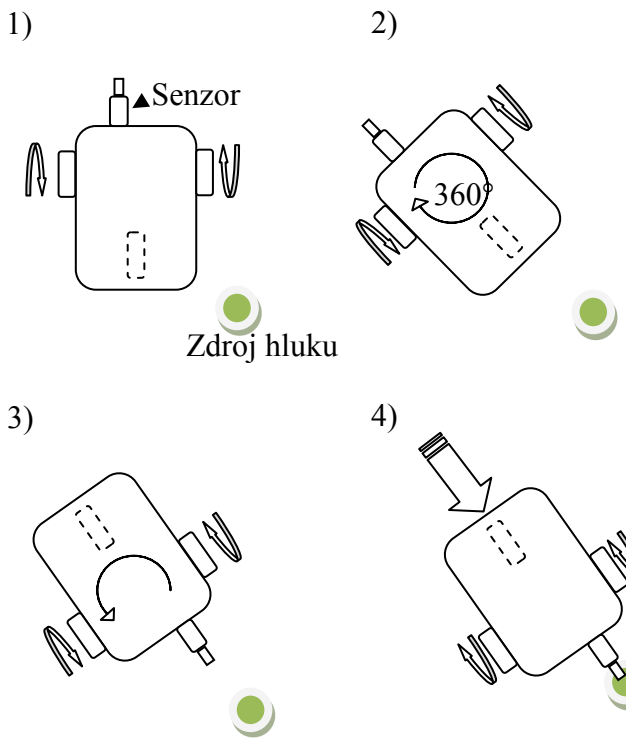
Po zvolení spuštění úlohy jsou na první řádek displeje vypsány informace o principu úlohy a požadovaném zapojení jednotlivých modulů. Zapojení senzoru zvuku je uvažováno prostřednictvím portu BAS2 a jednotlivé motory lze zapojit do libovolného portu pro motory sběrnice Bank A, kdy bude ovlivněn pouze směr otáčení soustavy při natáčení v prostoru. Po odsouhlasení těchto informací prostřednictvím stisknutí tlačítka GO jsou opět vypisovány na první řádek displeje informace o požadované akci uživatele. Nyní je potřeba umístit soustavu ke zdroji hluku tak, aby byl k němu zvukový senzor natočen. Po opětovném stisknutí tlačítka GO je naměřena referenční hladina intenzity zvuku v okamžiku stisku tlačítka a jsou vypsány další informace o požadované akci uživatele. V této fázi je potřeba umístit vozidlo dále od zdroje a poté je potřeba opětovně stisknout tlačítko GO. V tuto chvíli je zahájeno autonomní hledání zdroje při cyklickém

opakování tří kroků. V prvním kroku je soustava desetkrát natočena, kdy po každém natočení je naprogramováno zpoždění po zastavení motoru, protože servomotor za chodu generuje hluk o značné intenzitě a až po tomto zpoždění je naměřena aktuální hladina intenzity hluku v místě čidla zvukového senzoru. Po definovaném počtu natočení a měření hlukové intenzity nastává druhý krok a tím je vyhodnocení největší hlukové hladiny a natočení vlastní soustavy na směr, při kterém byla intenzita naměřena. Poté nastává poslední z těchto cyklických kroků, kdy jsou uvedeny do stavu chodu oba motory a soustava se mírně posune ve směru svého natočení. Pokud nebyla naměřena intenzita vetší nebo rovna referenční hladině zvuku, tak jsou zmíněné tři kroky opakovány tak dlouho, než je naměřená hladina rovna nebo větší hladině referenční. Pokud je dosáhnuto nalezení pozice zdroje hluku, stav úlohy je nastaven do fáze, kdy byla právě naměřena referenční hodnota intenzity a pro zahájení hledání je potřeba stisknout tlačítko GO. Pokud je požadavek na nekonečné hledání zdroje, je potřeba jako referenční hladinu nastavit takovou, kterou vozidlo nebude schopno nalézt a bude se posouvat za vždy největším nalezeným zdrojem hluku.

Algoritmus této úlohy je rozdělen celkem do šesti kroků, kdy algoritmy všech kroků jsou uvedeny postupně v rámci příloh H až L. Krok první je čistě informativním, kdy jsou nastaveny proměnné definující zapojení modulů, vypisovány informace o zapojení a je čekáno na stisk tlačítka GO, kdy je program přesunut do cyklického spouštění druhého kroku. V druhém kroku jsou také vypisovány informace uživateli a po stisku tlačítka GO je jednou načtena hodnota naměřená senzorem zvuku a uložena do proměnné představující referenční hodnotu intenzity hluku. Ve třetím kroku jsou vypisovány informace uživateli, kdy je potřeba soustavu oddálit od zdroje hluku a jsou inicializovány proměnné představující zpoždění před měřením, maximální naměřenou intenzitu při jedno cyklu otáčení, a hodnotu enkodéru před započetím otáčení.

Po stisknutí tlačítka GO je cyklicky spouštěn čtvrtý krok úlohy, kdy je prostřednictvím dvou parametrů vyhodnoceno, do jakého dalšího kroku je nutné úlohu posunout. První je kontrolována podmínka, jestli je dokončeno jedno natáčení před měřením, jako rozdíl aktuální hodnoty enkodéru a polohy před počátkem natáčení. Pokud ještě v daném okamžiku nebyla soustava natočena, motor je stále v chodu a tento krok je opakován. Pokud však podmínka neplatí a soustava je uvažována, jako natočená o požadovanou rozteč pro měření v tomto směru, je inkrementována hodnota proměnné definující počet natočení a úloha přechází do kroku pátého. Pokud je v rámci čtvrtého kroku hodnota definující počet natočení větší, než je nastavený celkový počet měření na otáčku, úloha je přesunuta do kroku šestého, soustava je natočena ve smyslu největší naměřené intenzity a je vynulována proměnná počtu natočení. Při opakovaném spouštění pátého kroku úlohy jsou prováděny akce týkající se vlastního měření intenzity zvuku. Nejprve je inkrementována proměnná definující počet provedených cyklů, později užitá pro realizaci zpoždění před vlastním měřením. Pokud ještě hodnota nebyla požadovaně inkrementována, jsou v tomto kroku pouze vypisovány informace uživateli a opakovaně dochází ke zvýšení hodnoty proměnné definující zpoždění. Pokud již bylo požadovaného zpoždění dosaženo, tak je naměřena hodnota intenzity hluku a porovnána s maximální

naměřenou hodnotou v rámci jednoho cyklu otáčení a s hodnotou referenční. Pokud je naměřená intenzita větší, než je hodnota referenční, úloha je přesunuta do kroku třetího. Pokud tak není a hodnota je pouze větší, než maximální naměřená intenzita v rámci otáčení, tak jsou uloženy hodnoty vlastní intenzity jako maximální na otočení a aktuální hodnota enkodéru. Po jednom provedení kroku pátého je aktuální hodnota natočení sestavy uložena jako počátek natočení enkodéru a přesunuta do kroku čtvrtého. Jak již bylo zmíněno, ve čtvrtém kroku úlohy je kontrolováno, zda již nebyla soustava natočena o jedno otočení kolem své osy a pokud tak je vyhodnoceno, soustava je natočena ve směru největší naměřené intenzity a přesouvá se do šestého kroku. Funkcí úlohy při nastaveném šestém kroku je posunout se ve směru největší naměřené intenzity hluku. Nejprve je tedy kontrolováno, zda je již soustava natočena v přípustném rozmezí požadovaného směru. Pokud ne, je odeslán požadavek na natočení enkodéru do pozice maximální naměřené hodnoty intenzity. Pokud již je vozidlo natočeno, oba motory jsou uvedeny do chodu s nastavenou maximální rychlostí, jsou vynulovány hodnoty proměnných představující natočení enkodéru při maximální naměřené intenzitě hluku, maximální naměřené hladiny hluku při otáčení a úloha je uvedena do kroku pátého.



Obr. 5.7: Schéma jednoho cyklu úlohy využívající senzor zvuku při hledání pozice zdroje hluku

Úloha jízdy po čáře

Principem této úlohy je, aby se vozidlo pohybovalo po nakreslené křivce o určité šířce. V rámci úlohy je použit jeden světelný senzor LEGO Mindstorms Education, který je provozován v režimu měření světelné intenzity. Senzor je namířen kolmo k povrchu, po kterém se vozidlo pohybuje a měří tedy intenzitu odraženého světela. Barvu čáry je potřeba volit tak, aby odrážela světlo o rozdílné intenzitě, než okolní plocha a byla možná indikace polohy senzoru nad čarou a mimo ní. Senzor musí být také vhodně vzdálen od měřeného povrchu tak, aby naměřené hodnoty co nejméně ovlivňovalo okolní světelné záření a aby nebylo měřené místo povrchu zastíněno senzorem a na čidlo tak mohlo dopadat odražené světlo od povrchu.

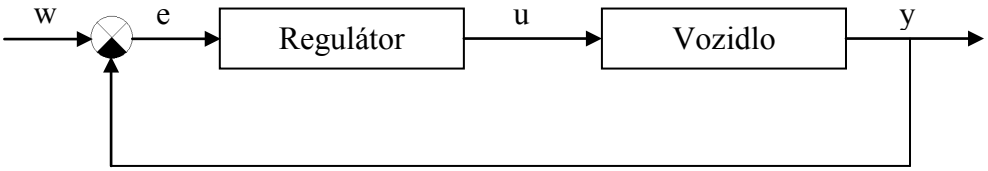
Prostřednictvím jednoho senzoru lze vyhodnotit, zda se nachází nad okolím křivky nebo ne. S použitím dvou senzorů, kdy jsou oba umístěny na rozdílné straně křivky a je známa jejich vzájemná poloha, lze také případně určit, na které ze stran vozidlo při pohybu vybočí. S použitím pouze jednoho senzoru nelze řídit pohyb po čáře, avšak pouze po její hraně, kdy při detekci polohy na čáře je vozidlo uvedeno do zatáčení do určité strany a pokud je detekována poloha mimo čáru, tak vozidlo zatáčí do strany druhé. Při detekci polohy mimo křivku tedy nelze určit, jestli je senzor od ní vpravo či vlevo. Při přejezdu vozidla mimo čáru na stranu, která není v rámci řízení uvažována, nastává poruchový stav a vozidlo je nutné manuálně umístit na stranu umožňující uvažované řízení.

Jelikož se jedná o úlohu, kde je potřeba rychlých reakcí na požadované změny směru natáčení, je vhodné pro tuto úlohu vytvořit nový časovač s periodou spouštění tak, aby byla odezva řízení na základě vyhodnocené poloze čáry dostatečná a vozidlo nepřejelo čáru na druhou stranu. V úloze je naprogramováno ovládání IR vysílačem, kdy je možné vozidlo uvést do chodu, zastavit ho a uvést vozidlo do režimu měření referenčních hodnot, ze kterých se v rámci programu odvíjí řídicí parametry. Tato fáze je schematicky znázorněna na obrázku 5.9.

Po spuštění úlohy je nejprve nutné vozidlo nechat naměřit referenční úrovně světelné intenzity odrážející se od povrchu čáry a jejího blízkého okolí. Navržený algoritmus je uveden v rámci přílohy M. Před spuštěním režimu je nutné umístit vozidlo na co nejpřímější část křivky a zaslat příkaz prostřednictvím stisknutí šipky dolů IR vysílače. Vozidlo se následně natočí a naměří potřebné informace v okolí křivky. Tímto procesem jsou uloženy dvě naměřené hodnoty, největší naměřená intenzita a nejmenší naměřená intenzita. Jako intenzita definující hraniční místo křivky, po které je potřeba řídit pohyb senzoru, je uvažován průměr těchto naměřených extrémních hodnot intenzit. V rámci vlastního řízení pohybu je pak naprogramována regulace na tuto průměrnou hodnotu.

Jako regulovaná žádaná veličina je tedy měřená intenzita, která je průměrem maximální a minimální naměřené intenzity v blízkém okolí křivky. Jako akční zásah je uvažována modifikace rychlostí hnacích kol, která má při vzájemně různých hodnotách za

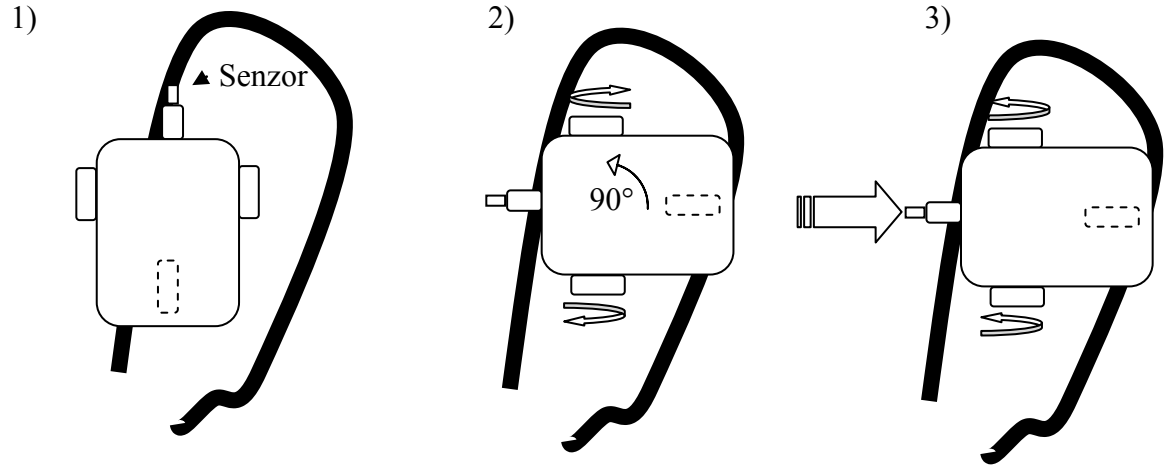
následek natáčení vozidla se senzorem. Jedná se tedy o řízení v otevřeném regulačním obvodu a jeho schéma je uvedeno v rámci obrázku 5.8.



Obr. 5.8: Schéma otevřeného regulačního obvodu

Výstupní veličinou je měřená světelná intenzita a cílem řízení je, aby se rovnala hodnotě žádané a regulační odchylka se tak blížila nule. Při rozdílných hodnotách naměřené a požadované hodnoty světelné intenzity je vyhodnocena nenulová regulační odchylka. Prostřednictvím průběhu regulační odchylky je pomocí naprogramovaného regulátoru vyhodnocován akční zásah v podobě modifikace rychlostí jednotlivých hnacích kol, což má za následek natáčení vozidla a senzoru nad křivkou. Jako vhodný typ regulátoru byl zvolen PD, který prostřednictvím nastavené hodnoty derivační složky reaguje rychle na výrazné změny regulační odchylky, což je vhodné pro sledování tvarů křivek, kde je nutné vyhodnotit ostré rychlé zatočení. Výpočet akčního zásahu pomocí PD regulátoru vyžaduje derivaci regulační odchylky, která je vypočtena jako rozdíl odchylky aktuální a předešlé.

Vozidlo se tedy pohybuje základní rychlostí, která je pro každé kolo regulována s ohledem na akční zásah. Jako základní rychlost každého kola je uvažováno 10% maximální nastavitelné rychlosti otáčení, kdy akční zásah je realizován v rámci přičtení jeho hodnoty k rychlosti otáčení prvního servomotoru a odečtení jeho hodnoty od nastavené rychlosti otáčení druhého servomotoru. Pokud se tedy naměřená intenzita zvyšuje nad úroveň požadované a regulační odchylka je záporná, znamená to, že senzor se nachází mimo křivku a je potřeba nastavit rychlosti motorů tak, aby byl smysl natáčení směrem ke křivce.



Obr. 5.9: Schéma natáčení vozidla při měření referenčních hodnot odražené světelné intenzity v okolí křivky

V rámci časovače, užívaného pro realizaci chodu předchozích úloh, s periodou spouštění 500 ms, je v úloze pohybu po čáře prováděno měření referenčních hodnot a vlastní řízení pohybu po hraně křivky je naprogramováno v rámci časovače s periodou 50ms. Po spuštění úlohy v rámci GUI je vozidlo v klidovém stavu a je očekáván příkaz prostřednictvím IR vysílače, po jehož indikaci je úloha převedena do dalšího kroku. V následujícím kroku nastává fáze nalezení extrémních hodnot světelné intenzity světla odraženého od povrchu v okolí křivky. Z těchto hodnot je vypočtena průměrná hodnota, která představuje předpokládanou naměřenou světelnou intenzitu na hraně křivky a je uložena jako požadovaná hodnota. Nyní je potřeba manuálně přemístit vozidlo čidlem na křivku a při indikaci stisknutí tlačítka IR vysílače nahoru nastává fáze řízení v rámci časovače s periodou 50ms. Pohyb je uvažován po pravé hraně křivky, pokud se tedy nachází vozidlo na vnější straně křivky, hodnota naměřené intenzity je vyšší, než požadovaná a regulační odchylka je záporná. Požadavek na směr pohybu vozidla je v tomto stavu doleva, levý servomotor se tedy musí pohybovat pomaleji, než servomotor pravý. K základní rychlosti levého motoru je tedy dle uvedených principů nutné polovinu hodnoty regulační odchylky přičíst a od rychlosti levého motoru polovinu hodnoty regulační odchylky odečíst. Navržený algoritmus realizující řízení je uveden v rámci přílohy N.

6 Závěr

V rámci diplomové práce je uveden rozbor metody nahrazení logické kostky stavebnice LEGO Mindstorms Education pomocí vývojové desky NXShield-Dx a jednodeskového počítače FEZ Cerbuino Bee. Na základě uvedených principů je navržen software s demonstrujícími úlohami, při kterých jsou užity moduly stavebnice LEGO Mindstorms a Gadgeteer moduly. Software je naprogramován v jazyce C# v rámci Gadgeteer projektu s využitím platforem .NET Micro Framework a .NET Gadgeteer.

V úvodu diplomové práce jsou uvedeny teoretické rozbory jednotlivých vývojových nástrojů a zařízení, které jsou v přímé návaznosti k tvorbě zmíněného software. Nejprve jsou blíže představeny použité vývojové platformy a jednodeskový počítač FEZ Cerbuino Bee. Následně je představena stavebnice LEGO Mindstorms Education, kdy jsou uvedeny základní i neoficiální metody programování logické kostky stavebnice. Logická kostka stavebnice, jež zároveň zastává funkci řídicí i programovatelné jednotky, je v uvažované aplikaci nahrazena vývojovou deskou NXShield-Dx.

Vývojová deska NXShield-Dx je řídicí jednotkou, která je navržena pro zapojení modulů stavebnice LEGO Mindstorms Education, avšak není schopna plnit funkce programovatelné jednotky. Při jejím teoretickém rozboru jsou uvedeny metody přístupu k registrům I^2C slave zařízení představující dvě sběrnice, jimiž je osazena vývojová deska NXShield-Dx s porty určenými pro zapojení zmíněných LEGO modulů. K hodnotám registrů je přistupováno prostřednictvím komunikačního rozhraní FEZ Cerbuino Bee v rámci pinů druhého Gadgeteer slotu. Vývojová deska byla navržena pro zapojení do jednodeskového počítače užitím Arduino kompatibilních sběrnic. V rámci tvorby software se nepodařilo zprostředkovat I^2C komunikaci s využitím vzájemného zapojení Arduino sběrnic a pro přenos komunikačních signálů byly na straně jednodeskového počítače použity piny Gadgeteer slotu I, podporujícího komunikaci na bázi protokolu I^2C. Pro komunikaci pomocí pinů Gadgeteer slotu je k dispozici pouze jeden komunikační kanál a při jedné variantě zapojení lze tedy přistupovat pouze k zařízením zapojeným do jedné sběrnice NXShield-Dx.

Před rozborem jednotlivých demonstrativních úloh jsou definovány funkce, které jsou v rámci programu využity pro čtení hodnot naměřených senzory a nastavení řízení servomotorů. Funkce umožňují nastavitelné zapojení senzorů do libovolného z podporovaných portů NXShield-Dx. Jako vykreslovací zařízení je použit GHI modul, dvouřádkový displej HD44780 a aby bylo možné během chodu programu zadávat uživatelské vstupy, jsou také deklarovány metody indikující stisknutí tlačítek, kterými je osazena deska NXShield-Dx. Prostřednictvím uživatelského rozhraní lze nejprve zvolit jednu z nahraných úloh a dále je před jejím spuštěním vyžadována volba zapojení potřebných modulů. Program je v libovolném okamžiku resetovatelný s naprogramovaným zastavením chodu všech probíhajících operací. V rámci diplomové práce byly vytvořeny demonstrativní úlohy, které využívají dva servomotory, senzory dotyku, světla, zvuku, GHI IR přijímač a displej HD44780. Software je také napsán s ohledem na možnost připojení dalších úloh, které je zprostředkováno jen prostřednictvím přepsání hodnot

proměnných definujících počet nahraných úloh a jejich názvy. Přidanou úlohu lze poté spustit v rámci volby úloh GUI. Pro možnost vypisování dlouhých textových řetězců byla vytvořena funkce, prostřednictvím které jsou informace pro zapsání na šestnáctiznakový řádek vhodně ořezávány a na displeji je takto možné zapisovat zdánlivě se posouvající text. Navržený software je napsán jako jeden soubor Program.cs v jazyce C#, který obsahuje všechny definované funkce pro komunikaci, GUI a naprogramované úlohy v podobě funkcí.

V rámci diplomové práce se podařilo zprovoznit komunikaci jednodeskového počítače FEZ Cerbuino Bee s vývojovou deskou NXShield-Dx pomocí Gadgeteer slotu typu I. Vývojová deska byla navržena pro zapojení v rámci rozhraní Arduino kompatibilních sběrnic, avšak zprostředkování protokolu I^2C není tímto rozhraním striktně omezeno. Značnou nevýhodou užití Gadgeteer slotu typu I pro komunikaci je ten, že zmíněný slot není dále k dispozici pro zapojení jiného GHI modulu a volné zůstanou pouze dva Gadgeteer sloty, ze kterých není ani jeden typu I, který vyžaduje pro svou funkci značné množství GHI modulů. Dalším záporem je možnost využití pouze jednoho komunikačního kanálu a lze tedy přistupovat pouze k registrům zařízení zapojených v portech osazených na jedné sběrnici NXShield-Dx. Jako používané piny Arduino sběrnic jsou tři analogové pro přenos digitálních signálů tlačítek a jeden pro uzemnění. Zbytek pinů těchto sběrnic jednodeskového počítače FEZ Cerbuino Bee je tedy možné použít pro přenos dalších signálů.

Software je navržen s ohledem na uživatelskou přístupnost a nezávislost na komunikaci s vývojovým prostředím prostřednictvím sériové komunikace. Při každém předpokládaném stavu programu jsou uživateli v rámci GUI vypisovány pravě probíhající akce i požadované vstupy a programový kód je opatřen poznámkami vysvětlující jednotlivé funkce programového kódu. V každém kroku lze program prostřednictvím dlouhého stisknutí tlačítka GO resetovat bez nutnosti restartovat jakékoliv zařízení a také zastavit vzdáleně chod motorů pomocí dálkového IR vysílače. Na program lze také navázat v rámci tvorby a přidání dalších úloh i nových řídicích funkcí s možností užití modulů LEGO Mindstorms Education a GHI Gadgeteer.

V rámci navrženého software je implementováno pět demonstrativních úloh, které je možné spustit prostřednictvím GUI a tlačítek NXShield-Dx. Pro některé úlohy je naprogramováno ovládání pouze IR vysílačem. Úlohy jsou navrženy tak, aby byly demonstrovány metody použití navržených funkcí pro řízení servomotorů a čtení naměřených dat senzory stavebnice LEGO Mindstorms Education. Prakticky je také předvedena realizace použití modulů LEGO Mindstorms Education a zároveň modulů GHI Gadgeteer.

7 Použité zdroje

ARDUINO. Arduino: LibraryTutorial [online]. 2007 [cit. 2014-06-29]. Dostupné z: http://arduino.cc/en/Hacking/LibraryTutorial

GHI ELECTRONICS LLC. FEZ Cerbuino BEE: GHI Electronics. [online]. 2013a [cit. 2014-06-29]. Dostupné z: https://www.ghielectronics.com/catalog/product/351

GHI ELECTRONICS LLC. Gadgeteer Sockets Quick Reference. [online]. 2013b [cit. 2014-06-29]. Dostupné z: https://www.ghielectronics.com/docs/305/gadgeteer-sockets-quick-reference

GHI ELECTRONICS LLC. Documents: I2C. [online]. 2015 [cit. 2014-06-29]. Dostupné z: https://www.ghielectronics.com/docs/12/i2c#71

GHI ELECTRONICS LLC. FEZ Cerbuino BEE: User Manual [online]. 2012 [cit. 2014-06-29]. Dostupné z: http://www.mouser.com/pdfdocs/GHI_Electronics_FEZ_Cerbuino_Bee.pdf

HANSEN, John. NBC: NBC_Guide [online]. 2011 [cit. 2015-04-26]. Dostupné z: http://bricxcc.sourceforge.net/nbc/doc/NBC_Guide.pdf

HURBAIN, Philo. LEGO EV3 and NXT hacks and robots [online]. 2000 [cit. 2014-06-29]. Dostupné z:http://www.philohome.com/nxt.htm

KELLY, James Floyd. The NXT STEP is EV3: Microsoft's Robotics Developer Studio Info. In: Mindstorms Blog [online]. 2010 [cit. 2015-04-26]. Dostupné z: http://www.thenxtstep.com/2010/07/microsofts-robotics-developer-studio.html

MICROSOFT CORPORATION. Overview: Microsoft Robotics. Developer network [online]. 2012 [cit. 2015-04-26]. Dostupné z: https://msdn.microsoft.com/en-us/library/bb483024.aspx

MICROSOFT OPEN TECHNOLOGIES. What is NETMF?. .NET Micro Framework [online]. 2011 [cit. 2015-05-03]. Dostupné z: http://www.netmf.com/what-is-netmf/

MICROSOFT. .NET Micro Framework Platform SDK. .NET Micro Framework Platform SDK [online]. 2015 [cit. 2015-04-23]. Dostupné z: https://msdn.microsoft.com/en-us/library/jj610646.aspx

MICROSOFT. Understanding .NET Micro Framework Architecture. Understanding .NET Micro Framework Architecture [online]. 2015 [cit. 2015-04-23]. Dostupné z: https://msdn.microsoft.com/en-us/library/jj646623%28v=vs.102%29.aspx

NXP SEMICONDUCTORS. UM10204: I2C-bus specification and user manual [online]. 2014 [cit. 2015-04-25]. Dostupné z: http://www.nxp.com/documents/user_manual/UM10204.pdf

OPENELECTRONS. Sourceforge.net: NXShield Arduino Library and Examples. [online]. 2014 [cit. 2015-05-6]. Dostupné z: http://sourceforge.net/projects/nxshield/files/

OPENELECTRONS. NXShield: Advanced developer guide. In: Openelectrons.com [online]. 2012 [cit. 2015-04-25]. Dostupné z: http://www.openelectrons.com/index.php?module=documents&JAS_DocumentManager_op=downloadFile&JAS_File_id=14

OPENELECTRONS. Openelectrons.com: NXShield User Guide and Documents. [online]. 2010 [cit. 2015-04-26]. Dostupné z: http://www.openelectrons.com/index.php?module=documents&JAS_DocumentManager_op=viewDocument&JAS_Document_id=1

THE LEGO GROUP. Customer Service: How is the EV3 different from the NXT? [online]. 2013-a [cit. 2015-04-25]. Dostupné z: http://service.lego.com/en-us/helptopics/products/themes/mindstorms/mindstorms-ev3/ev3-and-nxt-differences

THE LEGO GROUP. LEGO. EV3 User Guide [online]. 2013-b [cit. 2015-04-25]. Dostupné z: cache.lego.com/r/education/-/media/lego%20education/home/downloads/user%20guides/global/ev3/ev3-user-guide-en.pdf?l.r=-1623019953

THE LEGO GROUP. LEGO.com: Mindstorms Search Result. [online]. 2015 [cit. 2015-04-25]. Dostupné z: http://search-en.lego.com/?q=mindstorms&cc=US

8 Seznam příloh

Příloha A Zdrojový kód C# projektu .NET Gadgeteer, jenž je základem pro tvorbu GUI.

Příloha B Zdrojový kód C# projektu .NET Gadgeteer s vytvořeným uživatelským rozhraním, umožňující volbu úlohy a resetu GUI prostřednictvím tlačítek NXShield-Dx.

Příloha C Vývojový diagram se zdrojovým kódem algoritmu v jazyce C# pro volbu portu v rámci GUI

Příloha D Vývojový diagram se zdrojovým kódem algoritmu v jazyce C# pro vyhodnocení kolize

Příloha E Vývojový diagram se zdrojovým kódem algoritmu v jazyce C# realizující odjezd od kolize

Příloha F Vývojový diagram se zdrojovým kódem algoritmu v jazyce C# realizující řízení natočení ve směru vyhodnocené maximální dopadající světelné intenzity

Příloha G Vývojový diagram se zdrojovým kódem algoritmu v jazyce C# realizující měření maximální dopadající světelné intenzity

Příloha H Vývojový diagram se zdrojovým kódem algoritmu v jazyce C# realizující měření referenční úrovně zdroje hluku

Příloha I Vývojový diagram inicializace před hledáním zdroje hluku se zdrojovým kódem algoritmu v jazyce C#

Příloha J Vývojový diagram se zdrojovým kódem algoritmu v jazyce C# realizující natáčení soustavy při hledání směru dopadající největší hlukové intenzity

Příloha K Vývojový diagram se zdrojovým kódem algoritmu v jazyce C# realizující zpoždění před měřením intenzity hluku a porovnání s intenzitou hledaného zdroje

Příloha L Vývojový diagram se zdrojovým kódem algoritmu v jazyce C# realizující pojezd ve směru největší naměřené intenzity hluku

Příloha M Vývojový diagram se zdrojovým kódem algoritmu v jazyce C# realizující fázi měření referenčních hodnot světelné intenzity při úloze jízdy po čáře

Příloha N Vývojový diagram se zdrojovým kódem algoritmu v jazyce C# realizující řízení pohybu dle stanovené požadované hodnoty měřené světelné intenzity

Příloha CD-R V rámci diplomové práce je přiložen disk s navrženým software v podobě .NET Gadgeteer solution projektu, který obsahuje všechny funkce a úlohy uvedené v diplomové práci. Uložena je také elektronická kopie této diplomové práce.